A Framework for Reasoning About LF Specifications

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Mary Southern

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Gopalan Nadathur, Advisor
April, 2021

Acknowledgments

I'd like to thank my advisor, Gopalan Nadathur, without whose knowledge and support this dissertation would not have been completed. I would also like to thank my committee members, Eric Van Wyk, Andrew Gacek, and Paul Garrett, for the time and ideas they have shared in getting me to this point.

I am grateful for the support of my family, immediate and extended, for their invaluable material and emotional support. I want to especially thank my Mom for always being on my side and for making a bed available when I needed one most. Also my sister Anna Mary, who can get me to smile, is always up for being together, and keeps me curious; I hope you will be as proud of me as I am of you.

I'd also like to thank my friends, who have each in their way provided support and understanding. I am grateful in particular to Estelle Smith for her resilience and sound advice. I also want to thank Shelley Kandola and Maggie Ewing for bringing their knowledge to our study of Homotopy Type Theory and the friendship that continued after.

Roller derby has been a place that taught me how strong I can be, and gave me a space for self-reflection and understanding when I was lost. Though there are many whom I could name, I want to particularly acknowledge Syd, Ace, Robin, Patrick, and Amber for their friendship both within and outside derby.

The theater community has also given me a needed outlet and sense of self, and I want to especially thank Steph and Varghese Alexander who are both talented performers and warm friends.

Finally, I want to express my most enduring gratitude to Ted for his unending love, support, and squishes. He has brought comfort, confidence, inspiration, joy, and laughter each in their time, and is the one on whom I rely most. I could not have picked a better person to be stuck in quarantine with.

This dissertation is based on work supported by the National Science Foundation under Grant No. CCF-1617771. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Abstract

With growing reliance on software in the modern world there is also a growing interest in ensuring that these systems behave in desired ways. Many researchers are interested in using formal specifications to develop correct systems, relying on the specifications as a means for reasoning about properties of systems and their behaviour. In this thesis we focus on systems whose behaviour is described in a syntax-directed, rule-based fashion. Such systems are typically encoded through a description of the relevant objects of the system along with some relations between these objects defined through a collection of rules. Properties of such systems are expressed through these object relations, relating the validity of certain relations to the validity of others.

A specification language based on the dependently typed λ-calculus, the Edinburgh Logical Framework, or LF, is often used for specifying such systems. The objects of interest in the system are formalized through terms in the specification, and the dependencies permitted in types provide a natural means for formalizing the relationships between objects of the system. Under such an encoding, the terms inhabiting the dependent types of LF represent valid derivations of the relation in the system and thus reasoning about type inhabitation in LF will correspond to reasoning about the validity of relations in the system.

This thesis develops a framework for formalizing reasoning about specifications of systems written in LF, with the ultimate goal of formalizing the informal reasoning steps one would take in an LF setting. This formalization will center around the development of a reasoning logic that can express the sorts of properties which arise in reasoning about such specifications. In this logic, type inhabitation judgements in LF serve as atomic formulas, and quantification is permitted over both contexts and terms in these judgements. The logic permits arbitrary relations over derivations of LF judgements to be expressed using a collection of logical connectives, in contrast to other systems for reasoning about LF specifications. Defining a semantics for these formulas raises issues which we must address, such as how to interpret both term and context quantification as well as the relation between atomic formulas and the LF judgements they are meant to encode.

This thesis also develops a proof system which captures informal reasoning steps as sound inference rules for the logic. To achieve this we develop a collection of proof rules including mechanisms for both case analysis and inductive reasoning over the derivations of judgements in LF. The proof system also supports applying LF meta-theorems through proof rules that enforce the requirements of the LF meta-theorem that cannot be expressed in the logic.

We also implement a proof assistant called Adelfa that provides a means for mechanizing the approach to reasoning about specifications written in LF that is the subject of this thesis. A characteristic of this proof assistant is that it uses the proof rules that complement the logic to describe a collection of tactics that can be used to develop proofs in goal-driven fashion. One of the problems to be solved in this context is that of realizing the rule in the proof system that enables the analysis of LF typing judgements that appear as assumption formulas. We show that a form of unification called higher-order pattern unification can provide the basis for such a realization. The Adelfa system is used to develop a collection of examples which demonstrate the effectiveness of the framework and to showcase how informal reasoning about specifications written in LF can be formalized using the logic and associated proof system.

Contents

List of Figures **vi**

1 Introduction **1**
- 1.1 Specification and Reasoning about Systems . . . 2
- 1.2 Existing Approaches to Reasoning about LF Specifications . . . 4
- 1.3 The Approach to Reasoning Developed in this Thesis . . . 5
- 1.4 The Contributions of this Thesis . . . 7
- 1.5 Overview of the Thesis . . . 8

2 Canonical LF and the Specification of Object Systems **10**
- 2.1 Canonical LF . . . 10
  - 2.1.1 The Syntax . . . 11
  - 2.1.2 Simultaneous Hereditary Substitution . . . 12
  - 2.1.3 Wellformedness Judgements . . . 20
- 2.2 Formalizing Object Systems in LF . . . 23
- 2.3 Meta-Theoretic Properties of LF . . . 25

3 A Logic for Expressing Properties of LF Specifications **32**
- 3.1 The Formulas of the Logic . . . 34
- 3.2 The Interpretation of Formulas . . . 38
- 3.3 Understanding the Notion of Validity . . . 43
- 3.4 Nominal Constants and Invariance Under Permutations . . . 46

4 A Proof System for Constructing Arguments of Validity **49**
- 4.1 The Structure of Sequents . . . 50
- 4.2 The Core Proof Rules . . . 61
  - 4.2.1 Structural Rules . . . 62
  - 4.2.2 The Axiom and the Cut Rule . . . 66
  - 4.2.3 Rules for the Logical Symbols . . . 71
- 4.3 Proof Rules that Interpret Atomic Formulas . . . 82
  - 4.3.1 Analyzing an Atomic Assumption Formula with an Atomic Type . . 83
  - 4.3.2 Proof Rules that Introduce Atomic Formulas . . . 102
- 4.4 An Annotation Based Scheme for Induction . . . 108
  - 4.4.1 Extending Formula Syntax with Annotations . . . 109
  - 4.4.2 The Induction Proof Rule . . . 114
  - 4.4.3 Additional Proof Rules that Interpret Annotations . . . 119

4.5 Proof Rules Encoding LF Meta-Theorems 126

5 Adelfa: An Implementation of the Proof System 137
5.1 An Overview of the System 137
5.2 Finding Covering Sets of Solutions for Case Analysis 141
5.3 Focusing Formulas 145

6 Constructing Proofs Using Adelfa 148
6.1 Additive Identity for Natural Numbers 149
6.2 Type Uniqueness for the STLC 152
6.3 Admissibility of Cut for a Simple Sequent Calculus 156
6.4 The POPLmark Challenge 164

7 Comparisions with Related Work 168
7.1 Twelf 168
7.2 The Logic $\mathcal{M}_2^+$ 169
7.3 Beluga 170
7.4 Harpoon 170
7.5 Cocon 171
7.6 Abella-LF 171

8 Conclusion 173
8.1 Encoding Properties of Contexts in the Logic 174
8.2 Schematic Polymorphism 175

Bibliography 178

List of Figures

2.1 The Syntax of LF Expressions . . . 11
2.2 Applying Substitutions to Terms . . . 13
2.3 Applying Substitutions to Types . . . 14
2.4 Applying Substitutions to Contexts . . . 14
2.5 Arity Typing for Canonical Terms . . . 15
2.6 The Formation Rules for LF Signatures and Contexts . . . 20
2.7 The Formation Rules for LF Kinds, Types, and Terms . . . 21
2.8 An LF Specification for the Simply-Typed Lambda Calculus . . . 23

3.1 Terms and Types in the Logic . . . 34
3.2 Arity Kinding for Canonical Types . . . 35
3.3 Block Schemas and Context Schemas . . . 35
3.4 Wellformedness Judgements for Block and Context Schemas . . . 36
3.5 The Formulas of the Logic . . . 36
3.6 The Wellformedness Judgement for Formulas . . . 37
3.7 Instantiating a Context Schema . . . 39

4.1 Well-Formed Context Variable Types and their Instantiations . . . 51
4.2 The Structural Rules . . . 62
4.3 The Axiom and the Cut Rule . . . 68
4.4 The Logical Rules . . . 72
4.5 Proof Rules Interpreting Atomic Formulas . . . 103
4.6 The Induction Proof Rule . . . 115
4.7 Id Rule allowing Annotated Formulas . . . 122
4.8 Atomic Proof Rules allowing Annotated Formulas . . . 123
4.9 Rules Encoding Meta-Theoretic Properties of LF . . . 128

5.1 Tactics of Adelfa . . . 140
5.2 Extended Term Syntax . . . 143

6.1 An LF Specification for Natural Number Addition . . . 148
6.2 An LF Specification for the Simply-Typed Lambda Calculus . . . 152
6.3 A Simple Intuitionistic Sequent Calculus . . . 156
6.4 An LF Specification for Derivations . . . 158
6.5 LF Specification for System $F_{<:}$ . . . 165

Chapter 1

Introduction

With the proliferation of software in the modern world there is also a growing interest in ensuring that these artefacts will work as desired and, specifically, not cause harm to people or property. These notions can be formalized through the use of specifications, mathematical descriptions of the desired behaviour for a system. Many researchers and developers are interested in using formal specifications for various tasks related to the development of correct systems. Thus, such specifications have been used in the past as the basis for building prototypes of relevant systems and as a means for reasoning about deeper properties concerning the behaviour of the systems.

This thesis is motivated by the latter concern, i.e. reasoning about computational systems through their specifications. A key ingredient to the formal description of such systems is the language in which the specifications are presented. The focus in this thesis in on a particular specification language based on a dependently typed λ-calculus called the Edinburgh Logical Framework [HHP93] or LF. In the typical situation, the terms of the specification language are used to provide encodings of the objects that are of interest in the system being formalized. Dependent types, which effectively relate terms, then provide a useful and convenient means for encoding relationships between the objects of the system. This can lead to very natural encodings of rule-based systems where relations are captured as dependent types and the rules defining this relation become the constructors for expressions of this type. Under this interpretation, the terms of the LF specification represent valid derivations in the encoded system, and reasoning about the derivability of typing judgements in LF corresponds to reasoning about the validity of relations in the system.

In this thesis we present a framework for reasoning about systems through specifications written in LF. This framework comprises a logic we have developed for reasoning about such specifications, an associated proof system for formalizing the construction of proofs in this logic, and an implementation of the proof system mechanizing the construction of such proofs.

1.1 Specification and Reasoning about Systems

Our focus in this thesis is on reasoning about object systems that are described in a syntax-directed and rule-based fashion. In the specification of such systems, the syntactic structure of the expressions describing the objects of interest is used to present rules that define relations between the objects. The typing of terms in the simply-typed λ-calculus would be a system of this sort. As an illustration, we may consider a version of the calculus in which there a single base type, *unit*; other types are constructed using the function type constructor $\rightarrow$. The lambda terms are constructed from variables and the constant *unit* using applications and abstractions. An important relation in this context is that between lambda terms and types relative to an assignment of types to variables. Below we give rules that define this relation. Observe that these rules are driven by the syntax of lambda terms.

$$\frac{}{\Gamma \vdash \langle\rangle : \mathit{unit}} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau \vdash t : \tau'}{\Gamma \vdash \lambda x : \tau . t : \tau \rightarrow \tau'} \qquad \frac{\Gamma \vdash t_1 : \tau' \rightarrow \tau \quad \Gamma \vdash t_2 : \tau'}{\Gamma \vdash t_1\ t_2 : \tau}$$

A first requirement in transforming such presentations into a formal specification in LF is being able to represent relations. Use is made in this context of the fact that LF types can depend on LF terms. In particular, such *dependent types* are used to encode relations. For example suppose that we have described representations in LF for the types and terms in the simply typed λ-calculus.[1] Then, writing $\overline{e}$ to denote the LF representation of the

[1] An interesting aspect of such representations is the possibility of using the technique of higher-order abstract syntax or HOAS [MN87, PE88] in encoding abstraction in the object system by abstraction in the meta-language, i.e. LF. We elide a discussion of this aspect because it is orthogonal to our present focus.

object language expression e, the typing relation between a simply typed λ-calculus term t and a type τ can be encoded by a dependent type of the form $(\mathit{hastype}\ \bar{t}\ \bar{\tau})$; here, *hastype* is a type constant in LF that takes two terms as arguments and is designated to represent the typing relation in the object language. An interesting aspect of the LF calculus is that the rules describing the encoded relations may themselves be characterized by suitably typed LF constants that provide a means for constructing objects of the LF type encoding the conclusion relation of the rule. For example, consider the typing rule for an application term in the simply typed λ-calculus. Ignoring the typing context Γ which is treated implicitly via LF typing contexts in the standard encoding of λ-terms, we see that this rule has four schematic variables—t_1, t_2, τ and τ'—and two premises. Accordingly, the rule can be represented by a designated term-level constant *ofapp* in LF that takes six arguments whose types correspond to those of the representations of the four schematic variables and the two premises and that yields an object that has the type $(\mathit{hastype}\ \overline{(t_1\ t_2)}\ \bar{\tau})$.

Specifications developed in this way can be used to determine if particular relations hold between relevant objects in the object system. For example, assuming an encoding of the syntax and typing rules for the simply typed λ-calculus of the kind described above, we might want to determine if the object language $(\lambda x : \mathit{unit}.x)$ has the type $(\mathit{unit} \rightarrow \mathit{unit})$. Such a question translates into one about the *inhabitation* of a type relative to the LF specification. Thus, in the example under consideration, this becomes a question about whether there is an LF term of the type $(\mathit{hastype}\ \overline{(\lambda x : \mathit{unit}.x)}\ \overline{(\mathit{unit} \rightarrow \mathit{unit})})$ There is in fact such a term and it can be seen that that term will essentially encode a derivation of the typing judgement using the rules defining such judgements. More broadly, answering the inhabitation question mirrors a search for a derivation in the object system.

Specifications in LF provide the basis also for stating and reasoning about properties of an encoded system that are more general than simply verifying if a particular relation holds. For example, in the simply-typed λ-calculus an interesting property is that when a term is typeable then that type is unique, i.e. that if $t : \tau$ and $t : \tau'$ are both derivable, then τ and τ' must in fact be the same type. This property can be captured relative to the LF specification by an assertion that if the dependent types (*hastype* t τ) and (*hastype* t τ') are inhabited for any choice of term t, then it must be the cast that τ and τ' are the same. It is the statement of this more general form of properties and the process of reasoning about them that is the focus of this thesis.

1.2 Existing Approaches to Reasoning about LF Specifications

One approach to reasoning about specifications written in LF is to encode properties of the system also as (dependent) types. This approach is the basis of reasoning in the Twelf system [PS02] and it's related logic, $\mathcal{M}_2^+$ [Sch00]. Consider the property of type uniqueness; it is essentially a relation between terms of particular dependent types, and thus can itself be encoded as a dependent type in LF which takes these terms as parameters. To encode this property as a type we need first encode equality as an LF type *eq* taking two types as parameters, and encode that only identical types are equal via a constructor for the type (*eq* τ τ). Thus for type uniqueness we define a new dependent type *unique* which takes as parameters a term t, two types τ and τ', a term of type (*hastype* t τ), and a term of type (*hastype* t τ') and will return a term of type (*eq* τ τ'). A key to this approach to reasoning is the observation that if a function of this type were total, then one can conclude that the property it encodes holds of the specification, and thus the system. A "proof" in the Twelf system amounts to presenting a term of the described type and then demonstrating via an external process that that term is in fact total; in the example, that it works, no matter what actual term is chosen for t. The logic $\mathcal{M}_2^+$ provides a means for making explicit the reasoning steps used by the external process in Twelf.

The approach to reasoning that underlies the Twelf system has the drawback that, at the end of the process, it does not provide a witness in the form of a proof by which a conclusion was arrived at. The ideas underlying Twelf are also germane to the system Beluga [PD10]. While Beluga uses a richer type system based on Contextual Modal Type Theory [NPP08] to overcome some of the issues of expressivity with Twelf, it continues to have the basic limitation of Twelf described above. The logic $\mathcal{M}_2^+$ addresses this criticism.

However, it shares with Twelf and Beluga the limitation that the properties that can be expressed and reasoned about have a fundamentally functional structure, that is, quantifiers in them take the form of a prefix with a $\forall \ldots \exists \ldots$ structure.

A second approach to reasoning about specifications written in LF is to translate the specification into a predicate logic and use tools for reasoning about the translation to construct a proof. This approach was the motivation behind the the Abella-LF system [SC14]. The translation approach utilizes a connection between dependencies in types and relations between terms to map a given LF specification into one which is written in a predicate logic. The dependent types and terms of LF are translated into simple types and terms via erasure of dependencies; since this is clearly a lossy mapping the erased typing information is then encoded using predicates. For example, with the type uniqueness of the simply typed λ-calculus, the type *hastype* would be translated into a relation in predicate logic that relates a translated term with a translated type. The constructors for the dependent type are translated into clauses defining the relation via the same encoding. Due to the lossy nature of the term encoding, this translation is not one-to-one when the terms are not in a form which contains no β-redexes, or normal form. This is because the erasure of dependencies in the types of abstraction terms means there is a many to one mapping of such terms, and there will not be sufficient information to uniquely identify which of these terms is represented using only the translation of the LF type in the case of a non-canonical term structure. Further complicating this approach, it is unclear what proofs about the translated specification mean in the context of LF. Lifting of results proved in this way to LF (and hence the system of interest) is suspect without such understanding. A variant of this approach based on only canonical forms may address the former issue, but would not in itself address the latter.

1.3 The Approach to Reasoning Developed in this Thesis

We approach reasoning from the perspective of understanding LF and the derivability of judgements relative to a given LF specification. Our goal is to provide a formalization of informal reasoning as it is performed in the LF setting, and provide a reasoning logic which is based on understanding LF derivability. Our approach focuses on constructing proofs explicitly, using reasoning steps which naturally correspond to the structure of informal arguments about LF, rather than relying on external analyses. The work in this thesis would, for example, provide a theoretical basis for the translation approach to interpret reasoning steps done over the translation as LF reasoning steps and thus lift the proofs in a sensible way.

The sorts of formulas we are interested in will center around the LF typing judgements as atomic formulas, which are intended to represent the derivation of that judgement in LF. We will use the notation $\{G \vdash M : A\}$ to represent a derivation in LF that the term M inhabits the dependent type A under a context G. To express the relations over these derivations we permit formulas to be constructed using logical connectives as well as quantification over both the terms and contexts. For example, for the property of type uniqueness we would want to construct a formula which quantified over the term t, the types τ_1 and τ_2, and the LF terms inhabiting the types $(\mathit{hastype}\ t\ \tau_1)$ and $(\mathit{hastype}\ t\ \tau_2)$. Since the typing rules can also extend the context as we descend under bindings, the formula representing this property should also include a quantification over the LF context appearing in the atomic formulas. The result might be something of the form

$$\begin{aligned}&\forall t.\forall \tau_1.\forall \tau_2.\forall d_1.\forall d_2.\Pi\Gamma.\\ &\quad \{\Gamma \vdash d_1 : \mathit{hastype}\ t\ \tau_1\} \supset \{\Gamma \vdash d_2 : \mathit{hastype}\ t\ \tau_2\} \supset \exists d_3. \{\Gamma \vdash d_3 : \mathit{eq}\ \tau_1\ \tau_2\}.\end{aligned}$$

The design of such a logic raises some interesting questions.

We do not intend for the quantification over t, for example, to mean for every single possible term which can be constructed. On the other hand, attempting to restrict these terms using an LF type will be problematic; the occurrences of a quantified variable may be under different contexts or no context and it is unclear how such typing could be made sensible at the quantifier. Further, using LF types would inhibit the meaning of atomic formulas like $\{\Gamma \vdash d_1 : \mathit{hastype}\ t\ \tau_1\}$ as we would already have identified d_1 as inhabiting a particular LF type. Our approach addresses these issues by using simple arity types for quantifiers which capture the functional character of the terms.

We also must make sense of the quantification over contexts and its meaning. In our example property, the contexts which are relevant to reasoning are those whose bindings might play a role in determining the inhabitation of a *hastype* type, i.e. ones which are the encoding of the contexts of the object system. In this example, the system contexts consist only of bindings of the form $x : \tau$, and so we are interested in those LF contexts which are constructed using the bindings $(x : \mathit{term}, y : \mathit{hastype}\ x\ \tau)$ which encode this assumption. Unconstrained quantification would mean contexts of any form must be considered, which does not capture this notion. Further, such an interpretation leads to a system where analysis based on the derivability of typing judgements in LF would not be effective. To address this, we use the idea of context schemas to describe a regular structure for context variables and use these as types for the context quantifiers.

The questions we have discussed above pertain to the structure of the formulas in the logic and the semantics that governs them. A separate question is that of providing a basis for mechanized reasoning based on the semantics. In particular, we would desire to complement the description of the logic with a sound and effective proof system. While soundness is a question that can be settled theoretically, the demonstration of effectiveness requires an implementation and the experimentation with this implementation as well. Thus, these are also matters to be addressed in the successful development of the approach that we have described here.

1.4 The Contributions of this Thesis

There are three main contributions of this thesis. First is the definitions of a logic rooted in an understanding of reasoning about LF derivability. The atomic formulas of this logic represent LF derivations, and formulas are constructed to capture relations about derivability in LF. The logic consists of these formulas along with a semantics based on interpreting the derivability of judgements in LF.

Second is the development of a proof system for the logic which formalizes the construction of arguments of validity based on this semantics. Reasoning steps which are based on properties of LF derivability are encoded as proof rules in the proof system allowing, for informal arguments to be captured naturally as derivations.

Finally, we mechanize the construction of derivations in the proof system through the implementation of the Adelfa theorem prover. The effectiveness of this system is demonstrated through a collection of examples which have all been formalized in Adelfa.

1.5 Overview of the Thesis

In Chapter 2 we present the specifics of the specification language LF. In this thesis we choose to work with the Canonical LF [HL07] formulation which permits only canonical form terms to be typed. We will use the more detailed understanding of LF from this chapter to provide an overview of the structure reasoning about such specifications takes.

In Chapter 3 we present a logic for reasoning about LF derivability based on an interpretation of atomic formulas as derivations in LF. We describe the sorts of formulas of interest and provide their semantics which relies on checking the derivability of LF judgements.

In Chapter 4 we propose a proof system to formalize reasoning based on the logic. The proof rules in this system capture the sort of reasoning steps which are used in informal reasoning about the validity of formulas. The rules that we describe come in two forms: those that interpret the meanings of the logical connectives and those that build in an understanding of LF derivability that is embodied in atomic formulas.

In Chapter 5 we describe an implementation of the proof system that we call Adelfa. We describe how the system is used and specifics of how some of the more complex rules in the proof system are implemented.

In Chapter 6 we present a collection of examples which showcase the effectiveness of using Adelfa for reasoning about LF specifications. These examples cover a variety of different kinds of systems which have been specified using LF.

In Chapter 7 we contrast this work with previously developed approaches to reasoning about LF in more detail.

We conclude in Chapter 8 and include discussion of future avenues of work involving the logic.

Chapter 2

Canonical LF and the Specification of Object Systems

The methodology for modelling object systems in a specification language depends on there being a one-to-one correspondence between the objects to be described and the expressions that are used to describe them. The existence of such a correspondence is the substance of the so-called *adequacy theorems*. When LF is used as the specification language, the adequacy theorems typically rely on limiting attention to normal forms with respect to the β- and η-conversion rules in the λ-calculus; these normal forms are referred to as the *canonical terms* of the language. The original presentation of LF [HHP93] includes terms in both canonical and non-canonical form. Such a presentation simplifies the treatment of substitution but at the price of complicating arguments concerning adequacy and LF derivability. In light of this, an alternative treatment of LF has been proposed that admits only terms that are in β-normal form and that are well-typed only if they are additionally in η-long form [HL07, WCPW03]. We use this presentation of LF, called *canonical LF*, as the basis for this work. The first section recalls this presentation and develops notions related to it that will be used in the later parts of this thesis. Towards motivating the development of a reasoning logic, we then discuss the use of LF in representing object systems and in reasoning about them at an informal level. The chapter concludes with an identification of meta-theorems related to derivability in LF that are useful in informal arguments concerning this relation.

2.1 Canonical LF

Our presentation of canonical LF, henceforth referred to simply as LF, differs from that in [HL07] in two respects. First, we elide the subordination relation in typing judgements since it is orthogonal to the thrust of this thesis. Second, we treat substitution independently of LF typing judgements and we also extend the notion to include the simultaneous replacement of multiple variables. The elaboration below builds in these ideas.

$$
\begin{array}{rrcl}
\textbf{Kinds} & K & ::= & \mathrm{Type} \mid \Pi x{:}A.\, K \\[4pt]
\textbf{Canonical Type Families} & A, B & ::= & P \mid \Pi x{:}A.\, B \\
\textbf{Atomic Type Families} & P & ::= & a \mid P\ M \\[4pt]
\textbf{Canonical Terms} & M, N & ::= & R \mid \lambda x.\, M \\
\textbf{Atomic Terms} & R & ::= & c \mid x \mid R\ M \\[4pt]
\textbf{Signatures} & \Sigma & ::= & \cdot \mid \Sigma, c : A \mid \Sigma, a : K \\[4pt]
\textbf{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}
$$

Figure 2.1: The Syntax of LF Expressions

2.1.1 The Syntax

The syntax of LF expressions is described in Figure 2.1. The primary interest is in three categories of expressions: kinds, types which are indexed by kinds, and terms which are indexed by types. In these expressions, λ and Π are binding or abstraction operators. Relative to these operators, we assume the principle of equivalence under renaming that is applied as needed. We also assume as understood the notions of free and bound variables that are usual to expressions involving such operators. To ensure the absence of β-redexes, terms are stratified into *canonical* and *atomic* forms. A similar stratification is used with types that is exploited by the formation rules to force all well-typed terms to be in η-long form. We use x and y to represent term-level variables, which are bound by abstraction operators or in the contexts that are associated with terms. Further, we use c and d for term-level constants, and a and b for type-level constants, both of which are typed in signatures. The expression $A_1 \to A_2$ is used as an alternative notation for the type family $\Pi x{:}A_1.\, A_2$ when x does not appear free in A_2. An atomic term has the form $(h\ M_1\ \ldots\ M_n)$

where h is a variable or a constant. We refer to h as the *head symbol* of such a term.

2.1.2 Simultaneous Hereditary Substitution

We will need to consider substitution into LF expressions when explicating typing and other logical notions related to these expressions. To preserve the form of these expressions, it is necessary to build β-reduction into the application of such substitutions. An important consideration in this context is that substitution application must be a terminating operation. Towards ensuring this property, substitutions are indexed by types that are eventually intended to characterize the functional structure of expressions.

Definition 2.1 (Arity Types)**.** The collection of expressions that are obtained from the constant o using the binary infix constructor $\rightarrow$ constitute the *arity types*. Corresponding to each canonical type A, there is an arity type called its *erased form* and denoted by $(A)^-$ and given as follows: $(P)^- = o$ and $(\Pi x{:}A_1.\, A_2)^- = (A_1)^- \rightarrow (A_2)^-$.

Definition 2.2 (Substitutions)**.** A variable substitution θ is a finite set of tuples of the form $\{\langle x_1, M_1, \alpha_1\rangle, \ldots, \langle x_n, M_n, \alpha_n\rangle\}$, where, for $1 \leq i \leq n$, x_i is a distinct variable, M_i is a canonical term and α_i is an arity type.[1] Given such a substitution, $dom(\theta)$ denotes the set $\{x_1, \ldots, x_n\}$ and $rng(\theta)$ denotes the set $\{M_1, \ldots, M_n\}$.

Given a substitution θ and an expression E that is a kind, a type, a canonical term or a context, we wish the expression $E[\![\theta]\!]$ notionally to denote the application of θ to E. However, such an application is not guaranteed to exist. We therefore use the expression $E[\![\theta]\!] = E'$ to indicate when it is defined and has E' as a result. The key part of defining this relation is that of articulating its meaning when E is a canonical term. This is done in Figure 2.2 via rules for deriving this relation. These rules use an auxiliary definition of substitution into an atomic term which accounts for any normalization that is necessitated by the replacement of a variable by a term. The different categories of rules in this figure

[1] Note that by a systematic abuse of notation, n may be less than m in a sequence written in the form $s_m, \ldots, s_n$, in which case the empty sequence is denoted. In this particular instance, a substitution can be an empty set of triples.

are distinguished by being preceded by a box containing the judgement form they relate to. The extension of this definition to the case where E is a kind or a type corresponds essentially to the application of the substitution to the terms that appear within E. This idea is made explicit for types in Figure 2.3 and its elaboration for kinds is similar. A substitution is meaningfully applied to a context only when it does not replace variables to which the context assigns types and when a replacement does not lead to inadvertent capture. When these conditions are satisfied, the substitution distributes to the types that are assigned to the variables as the rules in Figure 2.4 make clear.

$$\boxed{M[\![\theta]\!] = M'}$$

$$\frac{R[\![\theta]\!]^r = R'}{R[\![\theta]\!] = R'} \qquad \frac{R[\![\theta]\!]^r = M' : \alpha'}{R[\![\theta]\!] = M'} \qquad \frac{x \text{ not free in } dom(\theta) \cup rng(\theta) \qquad M[\![\theta]\!] = M'}{(\lambda x.\, M)[\![\theta]\!] = \lambda x.\, M'}$$

$$\boxed{R[\![\theta]\!]^r = M' : \alpha'}$$

$$\frac{\langle x, M, \alpha\rangle \in \theta}{x[\![\theta]\!]^r = M : \alpha}$$

$$\frac{R[\![\theta]\!]^r = \lambda x.\, M' : \alpha' \to \alpha'' \qquad M[\![\theta]\!] = M'' \qquad M'[\![\{\langle x, M'', \alpha'\rangle\}]\!] = M'''}{(R\ M)[\![\theta]\!]^r = M''' : \alpha''}$$

$$\boxed{R[\![\theta]\!]^r = R'}$$

$$\frac{}{c[\![\theta]\!]^r = c} \qquad \frac{x \notin dom(\theta)}{x[\![\theta]\!]^r = x} \qquad \frac{R[\![\theta]\!]^r = R' \qquad M[\![\theta]\!] = M'}{(R\ M)[\![\theta]\!]^r = R'\ M'}$$

Figure 2.2: Applying Substitutions to Terms

We define a measure on substitutions that is useful in showing that their application terminates.

Definition 2.3 (Size)**.** The size of an arity type is the number of occurrences of $\to$ in it. The size of a substitution is the largest of the sizes of the arity types in each of its triples.

$$\frac{}{a\llbracket\theta\rrbracket = a} \qquad \frac{P\llbracket\theta\rrbracket = P' \quad M\llbracket\theta\rrbracket = M'}{(P\ M)\llbracket\theta\rrbracket = (P'\ M')}$$

$$\frac{x \text{ not free in } \mathit{dom}(\theta) \cup \mathit{rng}(\theta) \qquad A_1\llbracket\theta\rrbracket = A_1' \qquad A_2\llbracket\theta\rrbracket = A_2'}{(\Pi x{:}A_1.\, A_2)\llbracket\theta\rrbracket = \Pi x{:}A_1'.\, A_2'}$$

Figure 2.3: Applying Substitutions to Types

$$\frac{}{\cdot\llbracket\theta\rrbracket = \cdot} \qquad \frac{x \text{ not free in } \mathit{dom}(\theta) \cup \mathit{rng}(\theta) \qquad \Gamma\llbracket\theta\rrbracket = \Gamma' \qquad A\llbracket\theta\rrbracket = A'}{(\Gamma, x : A)\llbracket\theta\rrbracket = \Gamma', x : A'}$$

Figure 2.4: Applying Substitutions to Contexts

The following theorem shows that simultaneous hereditary substitution is terminating and the result will be unique if it exists.

Theorem 2.1 (Uniqueness)**.** *For any context, kind, type or canonical term E and any substitution θ, it is decidable whether there is an E' such that $E\llbracket\theta\rrbracket = E'$ is derivable. Moreover, there is at most one E' for which it is derivable. Similarly, for any atomic term R and substitution θ, it is decidable whether there is an R' or an M' and α' such that $R\llbracket\theta\rrbracket^r = R'$ or $R\llbracket\theta\rrbracket^r = M' : \alpha'$ is derivable. At most one of these judgements is derivable and for at most one R', respectively, M' and α'.*

Proof. This theorem is proved by induction first on the size of substitutions and then on the structure of expressions. We first prove it simultaneously for canonical and atomic terms, and then extended to atomic types, canonical types, kinds, and finally contexts. □

The following theorem shows that the application of a vacuous hereditary substitution always exists.

Theorem 2.2 (Vacuous Substitutions)**.** *If E is a kind, a type or a canonical term none of whose free variables is a member of $\mathrm{dom}(\theta)$, then $E\llbracket\theta\rrbracket = E$ has as derivation. If R is an atomic term none of whose free variables is a member of $\mathrm{dom}(\theta)$ then $R\llbracket\theta\rrbracket^r = R$ has a derivation.*

Proof. The proof is by induction on the structure of the expression. □

Simultaneous hereditary substitution enjoys a permutation property that is similar to the one described in [HL07] for unitary substitution. This is the content of the theorem below.

Theorem 2.3 (Permutation of Substitutions)**.** *Let θ_1 be an arbitrary substitution of the form $\{\langle x_1, M_1, \alpha_1\rangle, \ldots, \langle x_n, M_n, \alpha_n\rangle\}$. Further, let θ_2 be an arbitrary substitution of he form $\{\langle y_1, N_1, \beta_1\rangle, \ldots, \langle y_m, N_m, \beta_m\rangle\}$ where $y_1, \ldots, y_m$ are variables that are distinct from $x_1, \ldots, x_n$ and that do not appear free in $M_1, \ldots, M_n$. Finally, suppose that for each i, $1 \le i \le m$, there is some term N_i' such that $N_i[\![\theta_1]\!] = N_i'$ has a derivation and let θ_3 be the substitution defined by $\{\langle y_1, N_1', \beta_1\rangle, \ldots, \langle y_m, N_m', \beta_m\rangle\}$. For every kind, type, or canonical term E, E_1, and E_2 such that $E[\![\theta_1]\!] = E_1$ and $E[\![\theta_2]\!] = E_2$ have derivations, there must be an E' such that $E_2[\![\theta_1]\!] = E'$ and $E_1[\![\theta_3]\!] = E'$ have derivations.*

Proof. The proof proceeds by a primary induction on the sum of the sizes of θ_1 and θ_2 and a secondary induction on the derivation of $E[\![\theta_2]\!] = E_2$. We omit the details which are similar to those for Lemma 2.10 in [HL07]. □

$$\frac{c : \alpha \in \Theta}{\Theta \vdash^r_{at} c : \alpha} \qquad \frac{x : \alpha \in \Theta}{\Theta \vdash^r_{at} x : \alpha} \qquad \frac{\Theta \vdash^r_{at} R : \alpha' \to \alpha \qquad \Theta \vdash_{at} M : \alpha'}{\Theta \vdash^r_{at} R\ M : \alpha}$$

$$\frac{\{x : \alpha_1\} \uplus \Theta \vdash_{at} M : \alpha_2}{\Theta \vdash_{at} \lambda x.\, M : \alpha_1 \to \alpha_2} \qquad \frac{\Theta \vdash^r_{at} R : o}{\Theta \vdash_{at} R : o}$$

Figure 2.5: Arity Typing for Canonical Terms

While the application of a substitution to an LF expression may not always exist, this is guaranteed to be the case when certain arity typing constraints are satisfied as we describe below.

Definition 2.4 (Arity Typing)**.** An *arity context* Θ is a set of unique assignments of arity types to (term) constants and variables; these assignments are written as $x : \alpha$ or $c : \alpha$.

Given two arity contexts Θ_1 and Θ_2, we write $\Theta_1 \uplus \Theta_2$ to denote the collection of all the assignments in Θ_1 and the assignments in Θ_2 to the constants or variables not already assigned a type in Θ_1. The rules in Figure 2.5 define the arity typing relation denoted by $\Theta \vdash_{\mathrm{at}} M : \alpha$ between a term M and an arity type α relative to an arity context Θ. A kind or type E is said to respect an arity context Θ under the following conditions: if E is Type; if E is an atomic type and for each canonical term M appearing in E there is an arity type α such that $\Theta \vdash_{\mathrm{at}} M : \alpha$ is derivable; and if E has the form $\Pi x{:}A.\, E'$ and A respects Θ and E' respects $\{x : (A)^-\} \uplus \Theta$. A context Γ is said to respect Θ if for every $x : A$ appearing in Γ it is the case that A respects Θ. A substitution θ is *arity type preserving* with respect to Θ if for every $\langle x, M, \alpha \rangle \in \theta$ it is the case that $\Theta \vdash_{\mathrm{at}} M : \alpha$ is derivable. Associated with a substitution θ is the arity context $\{x : \alpha \mid \langle x, M, \alpha \rangle \in \theta\}$ that is denoted by $ctx(\theta)$.

Theorem 2.4 (Arity Type Preserving Substitution Always Defined)**.** *Let θ be a substitution that is arity type preserving with respect to Θ and let Θ' denote the arity context $\mathrm{ctx}(\theta) \uplus \Theta$.*

1. *If E is a canonical type or kind that respects the arity context Θ', then there must be an E' that respects Θ and that is such that $E[\![\theta]\!] = E'$ is derivable.*

2. *If M is a canonical term such that $\Theta' \vdash_{\mathrm{at}} M : \alpha$ is derivable, then there must be an M' such that $M[\![\theta]\!] = M'$ and $\Theta \vdash_{\mathrm{at}} M' : \alpha$ are derivable.*

3. *If R is an atomic term such that $\Theta' \vdash^r_{\mathrm{at}} R : \alpha$ is derivable, then either there is an atomic term R' such that $R[\![\theta]\!] = R'$ and $\Theta \vdash^r_{\mathrm{at}} R : \alpha$ are derivable or there is a canonical term M such that $R[\![\theta]\!] = M : \alpha$ and $\Theta \vdash_{\mathrm{at}} M : \alpha$ are derivable.*

Proof. The first clause in the theorem is an easy consequence of the second. We prove clauses (2) and (3) simultaneously by induction first on the sizes of substitutions and then on the structure of terms. The argument proceeds by considering the cases for the term structure, first proving (3) and then using this in proving (2). □

We will often consider expressions and substitutions that satisfy the arity typing requirements of the theorem above, which then guarantees that the applications of the substitutions have results. We introduce a notation that is convenient in this situation: we will write $E[\![\theta]\!]$ to denote the unique E' such that $E[\![\theta]\!] = E'$ has a derivation whenever such a derivation is known to exist.

Definition 2.5 (Composition of Substitutions)**.** Two substitutions θ_1 and θ_2 are said to be *arity type compatible* relative to the arity context Θ if θ_2 is type preserving with respect to Θ and θ_1 is type preserving with respect to $ctx(\theta_2) \uplus \Theta$. The composition of two such substitutions, written as $\theta_2 \circ \theta_1$, is the substitution

$$\{\langle x, M', \alpha\rangle \mid \langle x, M, \alpha\rangle \in \theta_1 \text{ and } M[\![\theta_2]\!] = M' \text{ has a derivation}\} \cup \\ \{\langle y, N, \beta\rangle \mid \langle y, N, \beta\rangle \in \theta_2 \text{ and } y \notin dom(\theta_1)\}$$

By Theorem 2.4 there must be an M' for which $M[\![\theta_2]\!] = M'$ has a derivation for each $\langle x, M, \alpha\rangle \in \theta_1$. Moreover such an M' must be unique. Thus, the composition described herein is well-defined. Note also that the composition must also be arity type preserving with respect to Θ.

Theorem 2.5 (Composition)**.** *Let θ_1 and θ_2 be substitutions that are arity type compatible relative to Θ and let Θ' denote the arity context* $\mathrm{ctx}(\theta_2 \circ \theta_1) \uplus \Theta$.

1. *If E is a canonical kind, type or context that respects Θ' and E' and E'' are, respectively, canonical types or kinds such that $E[\![\theta_1]\!] = E'$ and $E'[\![\theta_2]\!] = E''$ have derivations, then $E[\![\theta_2 \circ \theta_1]\!] = E''$ has a derivation.*

2. *If M is a canonical term such that, for some arity type α, $\Theta' \vdash_{\mathrm{at}} M : \alpha$ is derivable and M' and M'' are canonical terms such that $M[\![\theta_1]\!] = M'$ and $M'[\![\theta_2]\!] = M''$ have derivations, then $M[\![\theta_2 \circ \theta_1]\!] = M''$ has a derivation.*

3. *If R is a canonical term such that, for some arity type α, $\Theta' \vdash_{\mathrm{at}} R : \alpha$ is derivable and*

   (a) *M' and M'' are canonical terms such that $R[\![\theta_1]\!]^r = M' : \alpha$ and $M'[\![\theta_2]\!] = M''$ have derivations, then $R[\![\theta_2 \circ \theta_1]\!]^r = M'' : \alpha$ has a derivation;*

(b) *R' and M'' are, respectively, an atomic and a canonical term such that both $R[\![\theta_1]\!]^r = R'$ and $R'[\![\theta_2]\!]^r = M'' : \alpha$ have derivations then $R[\![\theta_2 \circ \theta_1]\!]^r = M'' : \alpha$ has a derivation;*

(c) *R' and R'' are atomic terms such that $R[\![\theta_1]\!]^r = R'$ and $R'[\![\theta_2]\!]^r = R''$ have derivations, then $R[\![\theta_2 \circ \theta_2]\!]^r = R''$ has a derivation.*

Proof. Clause 1 of the theorem follows easily from an induction on the structure of the canonical type or kind, assuming the property stated in clause 2.

We prove clauses 2 and 3 together. These clauses are premised on the existence of a derivation corresponding to the application of the substitution θ_1 to either M or R. The argument is by induction on the size of this derivation and it proceeds by considering the cases for the last rule in the derivation.

We consider first the cases where the derivation is for $M[\![\theta_1]\!] = M'$; the clause in the theorem relevant to these cases is 2. An easy argument using the induction hypothesis yields the desired conclusion when M is of the form $\lambda x. M_1$. In the case that M is an atomic term, there is a shorter derivation for $M[\![\theta_1]\!]^r = M' : \alpha$ or $M[\![\theta_1]\!]^r = M'$. In the first case, the induction hypothesis, specifically clause 3(a), allows us to conclude that $M[\![\theta_2 \circ \theta_1]\!] = M''$ has a derivation. In the second case, M' must be an atomic term and there must therefore be a derivation for $M'[\![\theta_2]\!]^r = M'' : \alpha$ or $M'[\![\theta_2]\!]^r = M''$. Using the induction hypothesis, specifically clause 3(b) or 3(c), we can again conclude that there must be a derivation for $M[\![\theta_2 \circ \theta_1]\!] = M''$.

We consider next the cases for the last rule when the derivation is for $R[\![\theta_1]\!]^r = M' : \alpha$.

- If M is a variable x such that $\langle x, M', \alpha\rangle \in \theta_1$, then it must be the case that there is some $\langle x, M'', \alpha\rangle \in \theta_2 \circ \theta_1$. Hence there must be a derivation for $M[\![\theta_2 \circ \theta_1]\!]^r = M'' : \alpha$.

- Otherwise M must be of the form $(R_1\ M_2)$ where there are derivations of judgements $R_1[\![\theta_1]\!] = \lambda x. M_3 : \alpha' \to \alpha$, $M_2[\![\theta_1]\!] = M_4$, and $M_3[\![\{\langle x, M_4, \alpha'\rangle\}]\!] = M'$ for suitable choices for M_3, α' and M_4. We note first that the arity context $(ctx(\theta_2 \circ \theta_1)) \uplus \Theta$

is equal to $ctx(\theta_1) \uplus (ctx(\theta_2) \uplus \Theta)$. Then, by the assumptions of the theorem and Theorem 2.4, it follows that there must be terms M_3' and M_4' such that $M_3[\![\theta_2]\!] = M_3'$ and $M_4[\![\theta_2]\!] = M_4'$ have derivations. We see by using the induction hypothesis with respect to the derivation for $R_1[\![\theta_1]\!] = \lambda x.\, M_3 : \alpha' \to \alpha$ that there must be a derivation for $R_1[\![\theta_2 \circ \theta_1]\!] = \lambda x.\, M_3' : \alpha' \to \alpha$. Using the induction hypothesis again with respect to the derivation for $M_2[\![\theta_1]\!] = M_4$, we see that there must be a derivation for $M_2[\![\theta_2 \circ \theta_1]\!] = M_4'$. By Theorem 2.3 it follows that $M_3'[\![\{\langle x, M_4', \alpha' \rangle\}]\!] = M''$ has a derivation and, hence that $(R_1\ M_2)[\![\theta_2 \circ \theta_1]\!]^r = M'' : \alpha$ has one too.

Finally we consider the cases for the last rule when the derivation is for $R[\![\theta_1]\!]^r = R'$. The argument when R is a constant is trivial. The case when R is a variable follows almost as immediately using the definition of $\theta_2 \circ \theta_1$. The only remaining case is when R is of the form $(R_1\ M_2)$ and R' is $(R_1'\ M_2')$ where $R_1[\![\theta_1]\!]^r = R_1'$ and $M_2[\![\theta_1]\!] = M_2'$ have shorter derivations for suitable terms R_1' and M_2'. We then have two subcases to consider with respect to the application of θ_2 to $(R_1'\ M_2')$:

- There is a derivation for $(R_1'\ M_2')[\![\theta_2]\!]^r = (R_1''\ M_2'')$ where R_1'' and M_2'' are terms such that $R_1'[\![\theta_2]\!]^r = R_1''$ and $M_2'[\![\theta_2]\!]^r = M_2''$ have derivations; note that the relevant clause in this case is 3(c) and R'' is $(R_1''\ M_2'')$. The induction hypothesis lets us conclude that $R_1[\![\theta_2 \circ \theta_1]\!]^r = R_1''$ and $M_2[\![\theta_2 \circ \theta_1]\!]^r = M_2''$ have derivations. Hence, $(R_1\ M_2)[\![\theta_2 \circ \theta_1]\!]^r = (R_1''\ M_2'')$ must have a derivation.

- There is a derivation for $(R_1'\ M_2')[\![\theta_2]\!]^r = M'' : \alpha$. In this case, for suitable choices for M_3, α' and M_4, there must be derivations for $R_1'[\![\theta_2]\!]^r = \lambda x.\, M_3 : \alpha' \to \alpha$, $M_2'[\![\theta_2]\!] = M_4$ and $M_3[\![\{\langle x, M_4, \alpha' \rangle\}]\!] = M''$. The induction hypothesis now lets us conclude that there are derivations for judgments $R_1[\![\theta_2 \circ \theta_1]\!]^r = \lambda x.\, M_3 : \alpha' \to \alpha$ and $M_2[\![\theta_2 \circ \theta_1]\!] = M_4$. It then follows easily that there must be a derivation for $(R_1\ M_2)[\![\theta_2 \circ \theta_1]\!]^r = M'' : \alpha$.

□

$\boxed{\vdash \Sigma\ \mathtt{sig}}$

$$\frac{}{\vdash \cdot\ \mathtt{sig}}\ \mathit{SIG\_EMPTY}$$

$$\frac{\vdash \Sigma\ \mathtt{sig} \qquad \cdot \vdash_\Sigma A\ \mathtt{type} \qquad c \text{ does not appear in } \Sigma}{\vdash \Sigma, c : A\ \mathtt{sig}}\ \mathit{SIG\_TERM}$$

$$\frac{\vdash \Sigma\ \mathtt{sig} \qquad \cdot \vdash_\Sigma K\ \mathtt{kind} \qquad a \text{ does not appear in } \Sigma}{\vdash \Sigma, a : K\ \mathtt{sig}}\ \mathit{SIG\_FAM}$$

$\boxed{\vdash_\Sigma \Gamma\ \mathtt{ctx}}$

$$\frac{}{\vdash_\Sigma \cdot\ \mathtt{ctx}}\ \mathit{CTX\_EMPTY}$$

$$\frac{\vdash_\Sigma \Gamma\ \mathtt{ctx} \qquad \Gamma \vdash_\Sigma A\ \mathtt{type} \qquad x \text{ does not appear free in } \Gamma}{\vdash_\Sigma \Gamma, x : A\ \mathtt{ctx}}\ \mathit{CTX\_TERM}$$

Figure 2.6: The Formation Rules for LF Signatures and Contexts

The erased form of a type is invariant under substitution. This is the content of the theorem below whose proof is straightforward.

Theorem 2.6 (Erasure is Invariant Under Substitution)**.** *For any type A and substitution θ, if $A[\![\theta]\!] = A'$ has a derivation, then $(A)^- = (A')^-$.*

Proof. The proof is by induction on the height of the derivation for $A[\![\theta]\!] = A'$ and using the definition of the erasure. □

2.1.3 Wellformedness Judgements

Canonical LF includes seven judgements: $\vdash \Sigma\ \mathtt{sig}$ that ensures that the constants declared in a signature are distinct and their type or kind classifiers are well-formed; $\vdash_\Sigma \Gamma\ \mathtt{ctx}$ that ensure that the variables declared in a signature are distinct and their type classifiers are well-formed in the preceding declarations and well-formed signature Σ; $\Gamma \vdash_\Sigma K\ \mathtt{kind}$ that determines that a kind K is well-formed with respect to a well-formed signature and context pair; $\Gamma \vdash_\Sigma A\ \mathtt{type}$ and $\Gamma \vdash_\Sigma P \Rightarrow K$ that check, respectively, the formation of

$\boxed{\Gamma \vdash_\Sigma K\ \texttt{kind}}$

$$\frac{}{\Gamma \vdash_\Sigma \mathrm{Type}\ \texttt{kind}}\ \mathit{CANON\_KIND\_TYPE}$$

$$\frac{\Gamma \vdash_\Sigma A\ \texttt{type} \qquad \Gamma, x : A \vdash_\Sigma K\ \texttt{kind}}{\Gamma \vdash_\Sigma \Pi x{:}A.\, K\ \texttt{kind}}\ \mathit{CANON\_KIND\_PI}$$

$\boxed{\Gamma \vdash_\Sigma A\ \texttt{type}}$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow \mathrm{Type}}{\Gamma \vdash_\Sigma P\ \texttt{type}}\ \mathit{CANON\_FAM\_ATOM}$$

$$\frac{\Gamma \vdash_\Sigma A_1\ \texttt{type} \qquad \Gamma, x : A_1 \vdash_\Sigma A_2\ \texttt{type}}{\Gamma \vdash_\Sigma \Pi x{:}A_1.\, A_2\ \texttt{type}}\ \mathit{CANON\_FAM\_PI}$$

$\boxed{\Gamma \vdash_\Sigma P \Rightarrow K}$

$$\frac{a : K \in \Sigma}{\Gamma \vdash_\Sigma a \Rightarrow K}\ \mathit{ATOM\_FAM\_CONST}$$

$$\frac{\Gamma \vdash_\Sigma P \Rightarrow \Pi x{:}A.\, K_1 \qquad \Gamma \vdash_\Sigma M \Leftarrow A \qquad K_1[\![\{\langle x, M, (A)^- \rangle\}]\!] = K}{\Gamma \vdash_\Sigma P\ M \Rightarrow K}\ \mathit{ATOM\_FAM\_APP}$$

$\boxed{\Gamma \vdash_\Sigma M \Leftarrow A}$

$$\frac{\Gamma \vdash_\Sigma R \Rightarrow P}{\Gamma \vdash_\Sigma R \Leftarrow P}\ \mathit{CANON\_TERM\_ATOM} \qquad \frac{\Gamma, x : A_1 \vdash_\Sigma M \Leftarrow A_2}{\Gamma \vdash_\Sigma \lambda x.\, M \Leftarrow \Pi x{:}A_1.\, A_2}\ \mathit{CANON\_TERM\_LAM}$$

$\boxed{\Gamma \vdash_\Sigma R \Rightarrow A}$

$$\frac{x : A \in \Gamma}{\Gamma \vdash_\Sigma x \Rightarrow A}\ \mathit{ATOM\_TERM\_VAR} \qquad \frac{c : A \in \Sigma}{\Gamma \vdash_\Sigma c \Rightarrow A}\ \mathit{ATOM\_TERM\_CONST}$$

$$\frac{\Gamma \vdash_\Sigma R \Rightarrow \Pi x{:}A_1.\, A_2 \qquad \Gamma \vdash_\Sigma M \Leftarrow A_1 \qquad A_2[\![\{\langle x, M, (A_1)^- \rangle\}]\!] = A}{\Gamma \vdash_\Sigma R\ M \Rightarrow A}\ \mathit{ATOM\_TERM\_APP}$$

Figure 2.7: The Formation Rules for LF Kinds, Types, and Terms

a canonical and atomic type relative to a well-formed signature, context and kind triple; and $\Gamma \vdash_\Sigma M \Leftarrow A$ and $\Gamma \vdash_\Sigma R \Rightarrow A$ that ensure, respectively, that a canonical and atomic term are well-formed with respect to a well-formed signature, context and canonical type triple. Figure 2.6 presents the rules for deriving the first two of these judgements, and the remaining judgments are presented in Figure 2.7. In the rules *CANON_KIND_PI* and *CANON_TERM_LAM* we assume x to be a variable that does not appear free in Γ. The formation rule for type and term level application, i.e. *ATOM_FAM_APP* and *ATOM_TERM_APP*, require the substitution of a term into a kind or a type. Use is made towards this end of hereditary substitution. The index for such a substitution is obtained by erasure from the type established for the term.

The judgement forms other than $\vdash \Sigma$ `sig` that are described above are parameterized by a signature that remains unchanged in the course of their derivation. In the rest of this thesis we will assume a fixed signature that has in fact been verified to be well-formed at the outset. The judgement forms require some of their other components to satisfy additional restrictions. For example, judgements of the form $\Gamma \vdash_\Sigma M \Leftarrow A$ require that Σ, Γ and A be well-formed as an ensemble. Judgements of the form $\Gamma \vdash_\Sigma R \Rightarrow A$ instead require that Σ and Γ be well-formed and ensure the well-formedness of both R and A. To be coherent, the rules in Figure 2.7 must ensure that in deriving a judgement that satisfies these requirements, it is necessary only to consider the derivation of judgements that also accord with these requirements. The fact that they possess this property can be verified by an inspection of their structure, using the observation that will be made in Theorem 2.11 that hereditary substitution preserves the property of being well-formed for kinds and types.

Arity typing judgements for terms approximate LF typing judgements as made precise below.

Definition 2.6 (Induced Arity Context)**.** The arity context induced by the signature Σ and context Γ is the collection of assignments that includes $x : (A)^-$ for each $x : A \in \Gamma$ and $c : (A)^-$ for each $c : A \in \Sigma$. When the context Γ is irrelevant or empty, we shall refer to the arity context as the one induced by just Σ.

$$
\begin{array}{ll}
\mathit{tp} : \mathrm{Type} & \mathit{of\_empty} : \mathit{of}\ \mathit{empty}\ \mathit{unit} \\
\mathit{unit} : \mathit{tp} & \\
\mathit{arr} : \mathit{tp} \to \mathit{tp} & \mathit{of\_app} : \Pi E_1{:}\mathit{tm}.\, \Pi E_2{:}\mathit{tm}.\, \Pi T_1{:}\mathit{tp}.\, \Pi T_2{:}\mathit{tp}. \\
 & \quad \Pi D_1{:}\mathit{of}\ E_1\ (\mathit{arr}\ T_1\ T_2).\, \Pi D_2{:}\mathit{of}\ E_2\ T_1. \\
\mathit{tm} : \mathrm{Type} & \quad \mathit{of}\ (\mathit{app}\ E_1\ E_2)\ T_2 \\
\mathit{empty} : \mathit{tm} & \\
\mathit{app} : \mathit{tm} \to \mathit{tm} \to \mathit{tm} & \mathit{of\_lam} : \Pi R{:}\mathit{tm} \to \mathit{tm}.\, \Pi T_1{:}\mathit{tp}.\, \Pi T_2{:}\mathit{tp}. \\
\mathit{lam} : \mathit{tp} \to (\mathit{tm} \to \mathit{tm}) \to \mathit{tm} & \quad \Pi D{:}(\Pi x{:}\mathit{tm}.\, \Pi y{:}\mathit{of}\ x\ T_1.\, \mathit{of}\ (R\ x)\ T_2). \\
 & \quad \mathit{of}\ (\mathit{lam}\ T_1\ (\lambda x.\, R\ x))\ (\mathit{arr}\ T_1\ T_2) \\
\mathit{of} : \mathit{tm} \to \mathit{tp} \to \mathrm{Type} & \\
\mathit{eq} : \mathit{tp} \to \mathit{tp} \to \mathrm{Type} & \mathit{refl} : \Pi T{:}\mathit{tp}.\, \mathit{eq}\ T\ T
\end{array}
$$

Figure 2.8: An LF Specification for the Simply-Typed Lambda Calculus

Theorem 2.7 (Arity Typing Approximates LF Typing)**.** *Let Θ be the arity context induced by the signature Σ and context Γ. If $\vdash_\Sigma \Gamma$ `ctx` then Γ respects Θ. If $\Gamma \vdash_\Sigma K$ `kind` or $\Gamma \vdash_\Sigma A$ `type` then, respectively, K or A respect Θ. If $\Gamma \vdash_\Sigma M \Leftarrow A$ is derivable, then $\Theta \vdash_{\mathrm{at}} M : (A)^-$ must also be derivable. If $\Gamma \vdash_\Sigma R \Rightarrow A$ is derivable, then $\Theta \vdash_{\mathrm{at}} R : (A)^-$ must also be derivable.*

Proof. The last two parts of the theorem are proved simultaneously by induction on the size of the derivation of $\Gamma \vdash_\Sigma M \Leftarrow A$ and $\Gamma \vdash_\Sigma R \Rightarrow A$. The first two parts follows from them, again by induction on the derivation size. □

2.2 Formalizing Object Systems in LF

A key use of LF is in formalizing systems that are described through relations between objects that are specified through a collection of inference rules. In the paradigmatic approach, each such relation is represented by a dependent type whose term arguments are encodings of objects that might be in the relation in question. The inference rules translate in this context into term constructors for the type representing the relation. We illustrate these ideas through an encoding of the typing relation for the simply-typed λ-calculus, a running example for this thesis.

We assume the reader to be familiar with the types and terms in the simply typed λ-calculus and also with the rules that define its typing relation. Figure 2.8 presents an LF signature that serves as an encoding of this system. This encoding uses the higher-order abstract syntax approach to treating binding. The specification introduces two type families, *tp* and *tm* to represent the simple types and λ-terms. Additionally, for each expression form in the object system, it includes a constant that produces a term of type *tp* or *tm*; as should be apparent from the declarations, we have assumed an object language whose terms are constructed from a single constant of atomic type that is represented by the LF constant *empty* and whose type is represented by the LF constant *unit*. This signature also provides a representation of two relations over object language expressions: typing between terms and types and equality between types. Specifically, the type-level constants *of* and *eq* are included towards this end. The rules defining the relations of interest in the object system are encoded by constants in the signature. The types associated with these constants ensure that well-formed terms of atomic type that are formed using the constants correspond to derivations of the relation in the object language that is represented by the type.

One of the purposes for constructing a specification is to use them to prove properties about the object system. For example, we may want to show that when a type can be associated with a term in the simply typed λ-calculus, it must be unique. Based on our encoding, this property can be stated as the following about typing derivations in LF:

> For any terms M_1, M_2, E, T_1, T_2, if there are LF derivations for $\vdash_\Sigma M_1 \Leftarrow \mathit{of}\, E\, T_1$ and $\vdash_\Sigma M_2 \Leftarrow \mathit{of}\ E\ T_2$, then there must be a term M_3 such than there is a derivation for $\vdash_\Sigma M_3 \Leftarrow \mathit{eq}\ T_1\ T_2$.

To prove this property, we would obviously need to unpack its logical structure. We would also need to utilize an understanding of LF in analyzing the hypothesized typing derivations. Considering the case where E is an abstraction will lead us to actually wanting to prove a more general property:

> For any terms M_1, M_2, E, T_1, T_2 and contexts Γ, if there are LF derivations for the judgements $\Gamma \vdash_\Sigma M_1 \Leftarrow \mathit{of}\ E\ T_1$ and $\Gamma \vdash_\Sigma M_2 \Leftarrow \mathit{of}\ E\ T_2$, then there must be a term M_3 such than there is a derivation for $\vdash_\Sigma M_3 \Leftarrow \mathit{eq}\ T_1\ T_2$.

Now, this property is not provable without some constraints on the form of contexts. In this example, it suffices to prove it when Γ is restricted to being of the form

$$(x_1 : \mathit{tm}, y_1 : \mathit{of}\ x_1\ Ty_1, \ldots, x_n : \mathit{tm}, y_n : \mathit{of}\ x_n\ Ty_n).$$

In completing the argument, we would need to use properties of LF derivability. A property that would be essential in this case is the finiteness of LF derivations, which enables us to use an inductive argument.

The objective in this thesis is to provide a formal mechanism for carrying out such analysis. We do this by describing a logic that is suitable for this purpose. One of the requirements of this logic is that it should permit the expression of the kinds of properties that arise in the process of reasoning. Beyond this, it should further be possible to complement the statement of properties with inference rules that permit the encoding of interesting and sound forms of reasoning.

2.3 Meta-Theoretic Properties of LF

Our reasoning system will need to embody an understanding of derivability in LF. We describe some properties related to this notion here that will be useful in this context. The first three theorems, which express structural properties about derivations, have easy proofs. The fourth theorem states a subsitutivity property for wellformedness judgements. This theorem is proved in [HL07].

Theorem 2.8. *If $\mathcal{D}$ is a derivation for $\Gamma \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma \vdash_\Sigma M \Leftarrow A$, then, for any variable x that is fresh to the judgement and for any A' such that $\Gamma \vdash_\Sigma A'$ $\texttt{type}$ is derivable, there is a derivation, respectively, for $\Gamma, x : A' \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma, x : A' \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma, x : A' \vdash_\Sigma M \Leftarrow A$ that has the same structure as $\mathcal{D}$.*

Theorem 2.9. *If $\mathcal{D}$ is a derivation for $\Gamma, x : A' \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma, x : A' \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma, x : A' \vdash_\Sigma M \Leftarrow A$ and x is a variable that does not appear free in K, A, or M and A respectively, then there must be a derivation that has the same structure as $\mathcal{D}$ for judgment $\Gamma \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma \vdash_\Sigma M \Leftarrow A$, respectively.*

Theorem 2.10. *If x does not appear in A_2 then $\Gamma_1, y : A_2, x : A_1, \Gamma_3$ is a well-formed context with respect to a signature Σ whenever $\Gamma_1, x : A_1, y : A_2, \Gamma_3$ is. Further, if there is a derivation $\mathcal{D}$ for $\Gamma, x : A_1, y : A_2, \Gamma_2 \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma, x : A_1, y : A_2, \Gamma_2 \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma, x : A_1, y : A_2, \Gamma_2 \vdash_\Sigma M \Leftarrow A$, then there must be a derivation that has the same structure as $\mathcal{D}$ for $\Gamma, y : A_2, x : A_1, \Gamma_2 \vdash_\Sigma K$ $\texttt{kind}$, $\Gamma, y : A_2, x : A_1, \Gamma_2 \vdash_\Sigma A$ $\texttt{type}$ or $\Gamma, y : A_2, x : A_1, \Gamma_2 \vdash_\Sigma M \Leftarrow A$, respectively.*

Theorem 2.11. *Assume that $\vdash_\Sigma \Gamma_1, x_0 : A_0, \Gamma_2$ $\texttt{ctx}$ and $\Gamma_1 \vdash_\Sigma M_0 \Leftarrow A_0$ have derivations, and let θ be the substitution $\{\langle x_0, M_0, (A_0)^- \rangle\}$. Then there is a Γ_2' such that $\Gamma_2[\![\theta]\!] = \Gamma_2'$ and $\vdash_\Sigma \Gamma_1, \Gamma_2'$ $\texttt{ctx}$ have derivations. Further,*

1. *if $\Gamma_1, x_0 : A_0, \Gamma_2 \vdash_\Sigma K$ $\texttt{kind}$ has a derivation, then there is a K' such that $K[\![\theta]\!] = K'$ and $\Gamma_1, \Gamma_2' \vdash_\Sigma K'$ $\texttt{kind}$ have derivations;*

2. *if $\Gamma_1, x_0 : A_0, \Gamma_2 \vdash_\Sigma A$ $\texttt{type}$ has a derivation, then there is an A' such that $A[\![\theta]\!] = A'$ and $\Gamma_1, \Gamma_2' \vdash_\Sigma A'$ $\texttt{type}$ have derivations; and*

3. *if $\Gamma_1, x_0 : A_0, \Gamma_2 \vdash_\Sigma M \Leftarrow A$ has a derivation (for some well-formed type A), there is an A' and an M' such that $A[\![\theta]\!] = A'$, $M[\![\theta]\!] = M'$, and $\Gamma_1, \Gamma_2' \vdash_\Sigma M' \Leftarrow A'$ have derivations.*

The reasoning system will need to build in a means for analyzing typing derivations of the form $\Gamma \vdash_\Sigma M \Leftarrow A$. This analysis will be driven by the structure of the type A. The decomposition when A is of the form $\Pi x_1{:}A_1.\, A_2$ has an obvious form. The development below, culminating in Theorem 2.12, provides the basis for the analysis when A is an atomic type.

Lemma 2.1. *Let Γ be a context such that $\vdash_\Sigma \Gamma$ $\texttt{ctx}$ has a derivation and let Θ be the arity context induced by Σ and Γ. Suppose that $\Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A$ is a type associated with a (term) constant or variable by Σ or Γ, or that $\Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, K$ is a kind associated with a (type) constant by Σ, where the y_is are distinct variables. Then, for $1 \leq i \leq n$, A_i and $\Pi y_i{:}A_i.\ \ldots \Pi y_n{:}A_n.\, A$ or, respectively, $\Pi y_i{:}A_i.\ \ldots \Pi y_n{:}A_n.\, K$ respect the arity context $\{y_1 : (A_1)^-, \ldots, y_{i-1} : (A_{i-1})^-\} \uplus \Theta$. Further, A or, respectively, K respects the arity context $\{y_1 : (A_1)^-, \ldots, y_n : (A_n)^-\} \uplus \Theta$.*

Proof. Since Σ and Γ are well-formed by assumption, depending on the case under consideration, either $\Gamma \vdash_\Sigma \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A$ $\texttt{type}$ or $\cdot \vdash_\Sigma \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, K$ $\texttt{kind}$ must have a derivation. The desired conclusions now follow from Theorem 2.7 and Definition 2.4. $\square$

Lemma 2.2. *Let Γ_1 be a context such that $\vdash_\Sigma \Gamma_1$ $\texttt{ctx}$ has a derivation, let Θ be the arity context induced by Σ and Γ_1, and let θ be a substitution that is arity type preserving with respect to Θ. Further, let x_0 be a variable that is neither bound in Γ_1 nor a member of $\mathrm{dom}(\theta)$, let A_0 and M_0 be such that $\Gamma_1 \vdash_\Sigma A_0$ $\texttt{type}$ and $\Gamma_1 \vdash_\Sigma M_0 \Leftarrow A_0$ are derivable and let $\theta' = \theta \cup \{\langle x_0, M_0, (A_0)^-\rangle\}$.*

1. *θ' is arity type preserving with respect to Θ.*

2. *Let Γ_2 be a context that respects an arity context Θ' such that $\mathrm{ctx}(\theta') \uplus \Theta \subseteq \Theta'$ and let Γ_2' be a context such that $\Gamma_2[\![\theta]\!] = \Gamma_2'$, and $\vdash_\Sigma \Gamma_1, x_0 : A_0, \Gamma_2'$ $\texttt{ctx}$ have derivations. Then there is a context Γ_2'' such that the following hold:*

   (a) $\Gamma_2'[\![\{\langle x_0, M_0, (A_0)^-\rangle\}]\!] = \Gamma_2''$, $\Gamma_2[\![\theta']\!] = \Gamma_2''$ and $\vdash_\Sigma \Gamma, \Gamma_2''$ $\texttt{ctx}$ have derivations;

   (b) if K is a kind that also respects Θ' and K' is a kind such that there are derivations for $K[\![\theta]\!] = K'$ and $\Gamma_1, x_0 : A_0, \Gamma_2' \vdash_\Sigma K'$ $\texttt{kind}$, then there is a kind K'' such that

$K'[\![\{\langle x_0, M_0, (A_0)^-\rangle\}]\!] = K''$, $K[\![\theta']\!] = K''$ *and* $\Gamma_1, \Gamma_2'' \vdash_\Sigma K''$ `kind` *are derivable; and*

(c) *if A is a type that also respects Θ' and A' is a type such that there are derivations for $A[\![\theta]\!] = A'$ and $\Gamma_1, x_0 : A_0, \Gamma_2' \vdash_\Sigma A'$* `type`*, then there is a type A'' such that $A'[\![\{\langle x_0, M_0, (A_0)^-\rangle\}]\!] = A''$, $A[\![\theta']\!] = A''$ and $\Gamma_1, \Gamma_2'' \vdash_\Sigma A''$* `type` *have derivations.*

Proof. Since $\Gamma_1 \vdash_\Sigma M_0 \Leftarrow A_0$ has a derivation, it follows from Theorem 2.7 that the substitution $\{\langle x_0, M_0, (A_0)^-\rangle\}$ is type preserving with respect to Θ. It then follows from the assumptions in the lemma that θ' is in fact $\{\langle x_0, M_0, (A_0)^-\rangle\} \circ \theta$ and type preserving with respect to Θ. The various observations in clause 2 now follow from Theorems 2.5 and 2.11. □

Theorem 2.12. *Let Γ be a context such that $\vdash_\Sigma \Gamma$* `ctx` *has a derivation.*

1. *$\Gamma \vdash_\Sigma R \Rightarrow A'$ has a derivation if*

   (a) *R is of the form $(c\ M_1\ \ldots\ M_n)$ for some $c : \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A \in \Sigma$ or of the form $(x\ M_1\ \ldots\ M_n)$ for some $x : \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A \in \Gamma$,*

   (b) *there is a sequence of types $A_1', \ldots, A_n'$ such that, for $1 \leq i \leq n$, there are derivations for both $A_i[\![\{\langle y_1, M_1, (A_1)^-\rangle, \ldots, \langle y_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!] = A_i'$ and $\Gamma \vdash_\Sigma M_i \Leftarrow A_i'$, and*

   (c) *$A[\![\{\langle y_1, M_1, (A_1)^-\rangle, \ldots, \langle y_n, M_n, (A_n)^-\rangle\}]\!] = A'$ and $\Gamma \vdash_\Sigma A'$* `type` *have derivations.*

2. *$\Gamma \vdash_\Sigma R \Rightarrow A'$ has a derivation of height h only if*

   (a) *R is of the form $(c\ M_1\ \ldots\ M_n)$ for some $c : \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A \in \Sigma$ or of the form $(x\ M_1\ \ldots\ M_n)$ for some $x : \Pi y_1{:}A_1.\ \ldots \Pi y_n{:}A_n.\, A \in \Gamma$,*

   (b) *there is a sequence of types $A_1', \ldots, A_n'$ such that, for $1 \leq i \leq n$, there is a derivation for $A_i[\![\{\langle y_1, M_1, (A_1)^-\rangle, \ldots, \langle y_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!] = A_i'$ and a derivation of height less than h for $\Gamma \vdash_\Sigma M_i \Leftarrow A_i'$, and*

(c) $A[\![\{\langle y_1, M_1, (A_1)^-\rangle, \ldots, \langle y_n, M_n, (A_n)^-\rangle\}]\!] = A'$ *and* $\Gamma \vdash_\Sigma A'$ $\texttt{type}$ *have derivations.*

Proof. At the outset, we should check the coherence of clauses 1(b) and 2(b) in the theorem statement by verifying that, for $1 \leq i \leq n$, it is the case that $\Gamma \vdash_\Sigma A'_i$ $\texttt{type}$ has a derivation. Towards this end, we first note that there must be a derivation for the type formation judgment $\Gamma, y_1 : A_1, \ldots, y_{i-1} : A_{i-1} \vdash_\Sigma A_i$ $\texttt{type}$ since Σ and Γ are well-formed. The desired conclusion then follows from using Lemma 2.2 repeatedly and observing, via Theorem 2.6, that erasure is preserved under substitution.

We now introduce some notation that will be useful in the arguments that follow. We will use Θ to denote the arity context induced by Σ and Γ. Further, for $1 \leq i \leq n+1$, we will write θ_i for the substitution $\{\langle y_1, M_1, (A_1)^-\rangle, \ldots, \langle y_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}$. An observation that we will make use of below is that if for $1 \leq j < i$ it is the case that $\Gamma \vdash_\Sigma M_i \Leftarrow A'_i$ has a derivation, then θ_i is type preserving with respect to Θ. This is an easy consequence of Theorems 2.7 and 2.6.

Proof of (1). We will consider explicitly only the case where R is $(c\ M_1\ \ldots\ M_n)$; the argument for the case when R is $(x\ M_1\ \ldots\ M_n)$ is similar. For $1 \leq i \leq n+1$ we will show that, under the conditions assumed for $M_1, \ldots, M_{i-1}$, there is a type A''_i such that $(\Pi y_i{:}A_i. \ldots \Pi x_n{:}A_n. A)[\![\theta_i]\!] = A''_i$, $\Gamma \vdash_\Sigma A''_i$ $\texttt{type}$ and $\Gamma \vdash_\Sigma (c\ M_1\ \ldots\ M_{i-1}) \Rightarrow A''_i$ have derivations. The desired conclusion follows from noting that A' must be A''_{n+1} because the result of substitution application is unique.

The claim is proved by induction on i. Consider first the case when i is 1. Since $\theta_1 = \emptyset$, A''_1 is $\Pi y_1{:}A_1. \ldots \Pi y_n{:}A_n. A$. The wellformedness of Σ ensures that $\Gamma \vdash_\Sigma A''_1$ $\texttt{type}$ has a derivation and we get a derivation for $\Gamma \vdash_\Sigma c \Rightarrow A'$ by using an *ATOM_TERM_CONST* rule.

Let us then assume the claim for i and show that it must also hold for $i+1$. By the hypothesis, there is an A''_i of the form $\Pi y_i{:}A'_i. A''$ where A'' is a type such that $(\Pi y_{i+1}{:}A_{i+1}. \ldots \Pi x_n{:}A_n. A)[\![\theta_i]\!] = A''$ has a derivation. Since $\Gamma \vdash_\Sigma A''_i$ $\texttt{type}$ has a derivation, so must $\Gamma, y_i : A'_i \vdash_\Sigma A''$ $\texttt{type}$. By Lemma 2.1, $\Pi y_{i+1}{:}A_{i+1}. \ldots \Pi x_n{:}A_n. A$ respects the arity context $\{y_1 : (A_1)^-, \ldots, y_i : (A_i)^-\} \uplus \Theta$. Since there are derivations for $\Gamma \vdash_\Sigma M_j \Leftarrow A'_j$ for $1 \leq j < i$, θ_i is type preserving over Θ. We now invoke Lemma 2.2 to conclude that there is a term A''' such that there are derivations for $A''[\![\{\langle y_i, M_i, (A'_i)^-\rangle\}]\!] = A'''$, $\Pi y_{i+1}{:}A_{i+1}. \ldots \Pi x_n{:}A_n. A[\![\theta_{i+1}]\!] = A'''$, and $\Gamma \vdash_\Sigma A'''$ $\mathtt{type}$. By the hypothesis, there is a derivation for $\Gamma \vdash_\Sigma c\, M_1\, \ldots\, M_{i-1} \Rightarrow \Pi y_i{:}A'_i. A''$. Using an *ATOM_TERM_APP* rule together with this derivation and the ones for $\Gamma \vdash_\Sigma M_i \Leftarrow A'_i$, and $A''[\![\{\langle y_i, M_i, (A'_i)^-\rangle\}]\!] = A'''$, we get a derivation for $\Gamma \vdash_\Sigma (c\ M_1\ \ldots\ M_i) \Rightarrow A'''$. Letting A''_{i+1} be A''' we see that all the requirements are satisfied.

Proof of (2). We prove the claim by induction on the height of the derivation of the type synthesis judgment $\Gamma \vdash_\Sigma R \Rightarrow A'$. We consider the cases for the last rule used in the derivation. If this rule is *ATOM_TERM_VAR* or *ATOM_TERM_CONST*, the argument is straightforward. The only case to be considered further, then, is that when the rule is *ATOM_TERM_APP*.

In this case, we know that R must be of the form $(R'\ M')$ where there is a shorter derivation for $\Gamma \vdash_\Sigma R' \Rightarrow B'$ for some type B'. From the induction hypothesis, it follows that R' has the form $(c\ M_1\ \ldots\ M_n)$ or $(x\ M_1\ \ldots\ M_n)$ for some $c : \Pi y_1{:}A_1. \ldots \Pi y_n{:}A_n. B \in \Sigma$ or $x : \Pi y_1{:}A_1. \ldots \Pi y_n{:}A_n. B \in \Gamma$ and that there must be a sequence of types $A'_1, \ldots, A'_n$ that, together with the terms $M_1, \ldots, M_n$ satisfy the requirements stated in clause 2(b). Moreover, B' must be such that $B[\![\theta_{n+1}]\!] = B'$ and $\Gamma \vdash_\Sigma B'$ $\mathtt{type}$ have derivations. Since the rule is an *ATOM_TERM_APP*, B' must have the structure of an abstracted type. From this it follows that B must be of the form $\Pi y_{n+1}{:}A_{n+1}. A$ and, correspondingly, B' must be of the form $\Pi y_{n+1}{:}A'_{n+1}. A''$ where $A_{n+1}[\![\theta_{n+1}]\!] = A'_{n+1}$ and $A[\![\theta_{n+1}]\!] = A''$ have derivations. Noting that the type of c or x is really of the form $\Pi y_1{:}A_1. \ldots \Pi y_{n+1}{:}A_{n+1}. A$ it follows from Lemma 2.1 that A respects the arity context $\{y_1 : (A_1)^-, \ldots, y_{n+1} : (A_{n+1})^-\} \uplus \Theta$. Also, since $\Gamma \vdash_\Sigma B'$ $\mathtt{type}$ has a derivation, it must be the case that $\Gamma, y_{n+1} : A'_{n+1} \vdash_\Sigma A''$ $\mathtt{type}$ has one. Since the derivation concludes with a *ATOM_TERM_APP* rule, it must be the case that $\Gamma \vdash_\Sigma M' \Leftarrow A'_{n+1}$ and $A''[\![\{\langle y_{n+1}, M', (A'_{n+1})^-\rangle\}]\!] = A'$ have shorter derivations than the one for $\Gamma \vdash_\Sigma R \Rightarrow A'$. Since θ_{n+1} is type preserving with respect to Θ, we may now use

Lemma 2.2 and Theorem 2.6 to conclude that $A[\![\theta_{n+1} \cup \{\langle y_{n+1}, M_2, (A_{n+1})^{-}\rangle\}]\!] = A'$ and $\Gamma \vdash_{\Sigma} A'$ $\texttt{type}$ have derivations. Renaming M' to M_{n+1} we see that all the requirements of clause 2 are satisfied. $\square$

Theorem 2.12 gives us an alternative means for deriving judgements of the analysis form $\Gamma \vdash_{\Sigma} R \Leftarrow P$, in the process dispensing with judgements of the synthesis form $\Gamma \vdash_{\Sigma} R \Rightarrow A$. Note also that in *analyzing* judgements of the form $\Gamma \vdash_{\Sigma} R \Leftarrow P$, it is necessary to consider only *shorter* derivations for subterms of R. This observation will be used in developing a means for arguing inductively on the heights of LF derivations.

A property similar to that in Theorem 2.12 can be observed for wellformedness judgements for atomic types. Theorem 2.13 presents a version that suffices for this thesis. A proof of this theorem can be constructed based essentially on the one for Theorem 2.12.

Theorem 2.13. *Let Γ be a context such that $\vdash_{\Sigma} \Gamma$ $\texttt{ctx}$ is derivable. Then $\Gamma \vdash_{\Sigma} P \Rightarrow K'$ has a derivation if and only if there is an $a : \Pi y_1{:}A_1. \ldots \Pi y_n{:}A_n. K \in \Sigma$ such that*

1. *P is of the form $(a\ M_1\ \ldots\ M_n)$;*
2. *there is a sequence of types $A'_1, \ldots, A'_n$ such that, for $1 \leq i \leq n$, there are derivations for $A_i[\![\{\langle y_1, M_1, (A_1)^{-}\rangle, \ldots, \langle y_{i-1}, M_{i-1}, (A_{i-1})^{-}\rangle\}]\!] = A'_i$ and $\Gamma \vdash_{\Sigma} M_i \Leftarrow A'_i$; and*
3. *$K[\![\{\langle y_1, M_1, (A_1)^{-}\rangle, \ldots, \langle y_n, M_n, (A_n)^{-}\rangle\}]\!] = K'$ and $\Gamma \vdash_{\Sigma} K'$ $\texttt{kind}$ have derivations.*

Chapter 3

A Logic for Expressing Properties of LF Specifications

Our objective in this chapter is to describe a logic in which we can express properties of an object system that has been specified in LF. The discussions in Section 2.2 suggest a possible structure for such a logic. The logic would be parameterized by an LF signature that has been determined to be well-formed at the outset. The basic building blocks for the properties that are to be described would be typing judgements. More specifically, the logic would use such judgements as its atomic formulas and would interpret them using LF derivability. More complex formulas would then be constructed using logical connectives and quantifiers over LF terms. As the example in Section 2.2 illustrates, it would be necessary to also permit a quantification over LF contexts.

To develop an actual logic based on these ideas, we need to describe a more precise correspondence between LF typing judgements and atomic formulas. The judgement forms that need to be considered in this context are those for typing canonical and atomic terms, i.e., the $\Gamma \vdash_\Sigma M \Leftarrow A$ and $\Gamma \vdash_\Sigma R \Rightarrow A$ forms. The main judgement form is in fact the first one: the second form serves mainly to explicate judgements of the first kind when the type is atomic and, as we have noted already, Theorem 2.12 provides the basis for circumventing such an explicit treatment through a special "focused" typing rule. In light of this, it suffices to describe an encoding of only the first judgement form. The judgement in the LF setting assumes the wellformedness of the context Γ and the type A. In the logic, the context and, therefore, also the type can be dynamically determined by instantiations for context variables. To deal with this situation, we will build the wellformedness of Γ and A into the interpretation of the encoding of the judgement. There is, however, an aspect of the wellformedness checking that we would like to extract into a static pre-processing phase.

The LF typing rules combine the checking of canonicity of terms with the determination of inhabitation that relies on the semantically more meaningful aspect of dependencies in types. To allow the focus in the logic to be on the latter aspect, we will build the former into a wellformedness criterion for formulas using arity types.

Another aspect that needs further consideration is the treatment of contexts in atomic formulas. To support typing derivations that use the *CANON_TERM_LAM* rule, such contexts must allow for the explicit association of types with variables. These variables may appear free in the terms and types in the atomic formula. However, their interpretation in this context must be different from the variables that are bound by quantifiers: in particular, these variables cannot be instantiated and each of them must be treated as being distinct within the atomic formula. The necessary treatment of these variables can be realized by representing them by *nominal constants* in the style of [GMN11, Tiu06]. Contexts must, in addition, allow for an unspecified part whose exact extent is to be determined by instantiation of an external context quantifier. To support this ability, we will allow context variables to appear in contexts. However, as observed in Section 2.2, we would like to be able to restrict the instantiation of such variables to blocks of declarations adhering to specified forms. To impose such constraints, the logic will permit context variables to be typed by *context schemas* that are motivated by regular world descriptions used in the Twelf system [PS02, Sch00].

In the rest of this chapter, we describe the logic in detail, thereby substantiating the ideas outlined above. The first two sections present the well-formed formulas and identify their intended meaning. The end result of this discussion is a means for describing properties of a specification comprised of an LF signature and for assessing the validity of such properties. The third section illuminates this capability through a collection of examples. The last section observes a property that will be useful in later chapters, namely, the irrelevance of the particular names that are chosen for the variables bound by the context in an LF judgement. Noting that these variables are represented by nominal constants in the logic, the statement of this property takes the form of invariance of validity of atomic formulas under permutations of nominal constants.

3.1 The Formulas of the Logic

$$
\begin{array}{rrcl}
\textbf{Terms} & M, N & ::= & R \mid \lambda x.\, M \\
\textbf{Atomic Terms} & R & ::= & c \mid x \mid n \mid R\ M \\
\textbf{Types} & A & ::= & P \mid \Pi x{:}A_1.\, A_2 \\
\textbf{Atomic Types} & P & ::= & a \mid P\ M
\end{array}
$$

Figure 3.1: Terms and Types in the Logic

We begin by considering the representation of LF terms and types in the logic. Figure 3.1 presents the syntax of the corresponding expressions. As with LF syntax, we use c and d to represent term level constants, a and b to represent type level constants and x and y to represent term-level variables. We also use n to represent a special category of symbols called the nominal constants. LF terms and types are obviously a subset of the expressions presented here. Going the other way, there are two main additions to the LF counterparts in the collection of expressions described here. First, nominal constants may be used in constructing terms. Second, as we shall soon see, variables may be bound not only by term and type level abstractions but also by formula level quantifiers.

We assume as given a set $\mathcal{N}$ of nominal constants, each specified with an arity type. Elements of $\mathcal{N}$ are written in the form $n : \alpha$. We assume that there is a countably infinite supply of nominal constants in $\mathcal{N}$ for each arity type α. The logic is parameterized by an LF style signature Σ that assigns kinds to type-level constants and types to term-level ones. This signature is assumed to be well-formed in the sense described in Section 2.1.

As explained earlier, expressions in the logic will be expected to satisfy typing constraints that check for canonicity. At the term level, these constraints will be realized through arity typing relative to a suitable arity context. At the type level, we must additionally ensure that (type) constants have been supplied with an adequate number of arguments. We make

$$\frac{a : K \in \Sigma}{\Theta \vdash^p_{ak} a : K} \qquad \frac{\Theta \vdash^p_{ak} P : \Pi x{:}A.\, K \qquad \Theta \vdash_{at} M : (A)^-}{\Theta \vdash^p_{ak} P\ M : K}$$

$$\frac{\Theta \vdash^p_{ak} P : \mathrm{Type}}{\Theta \vdash_{ak} P\ \mathit{type}} \qquad \frac{\Theta \vdash_{ak} A_1\ \mathit{type} \qquad \{x : (A_1)^-\} \uplus \Theta \vdash_{ak} A_2\ \mathit{type}}{\Theta \vdash_{ak} \Pi x{:}A_1.\, A_2\ \mathit{type}}$$

Figure 3.2: Arity Kinding for Canonical Types

these notions precise below; we assume the obvious extension of erasure to types in the logic here and elsewhere.

Definition 3.1. The typing relation between an arity context, a term and an arity type that is described in Definition 2.4 is extended to the present context by permitting terms to contain nominal constants and by allowing arity contexts to contain assignments to such constants. The rules in Figure 3.2 define an arity kinding property denoted by $\Theta \vdash_{ak} A\ \mathit{type}$ for a type A relative to an arity context Θ. In these rules, Σ is the signature parameterizing the logic. We will often need to refer to the arity context induced by Σ. We call this the *initial constant context* and we reserve the symbol Θ_0 to denote it.

Hereditary substitution extends naturally to the terms and types in the logic by treating nominal constants like other constants. The following theorem relating to such substitutions has an obvious proof.

Theorem 3.1. *If θ is type preserving with respect to Θ and $ctx(\theta) \uplus \Theta \vdash_{ak} A$ type and $A[\![\theta]\!] = A'$ have derivations, then $\Theta \vdash_{ak} A'$ type has a derivation.*

$$\begin{array}{rcll} \textbf{Block Declarations} & \Delta & ::= & \cdot \mid \Delta, y : A \\ \textbf{Block Schema} & \mathcal{B} & ::= & \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}\Delta \\ \textbf{Context Schema} & \mathcal{C} & ::= & \cdot \mid \mathcal{C}, \mathcal{B} \end{array}$$

Figure 3.3: Block Schemas and Context Schemas

The logic allows for quantifiers over contexts. In the intended interpretation, such quantifiers are meant to be instantiated with actual contexts that will correspond to assignments of LF types to nominal constants. However, it will be necessary to be able to constrain the possible instantiations in real applications. This ability is supported by typing context quantifiers using *context schemas* whose structure is presented in Figure 3.3. In essence, a context schema comprises a collection of *block schemas*. A block schema consists of a header of variables annotated with arity types and a body of declarations associating types with variables. Each variable in the header and that is assigned a type in the body of a block schema is required to be distinct. A block is intended to serve as a template for generating a sequence of bindings for nominal constants through an instantiation process that will be made clear in the next section. An actual context corresponding to a context schema is to be obtained by some number of instantiations of its block schemas. Block and context schemas are required to satisfy typing constraints towards ensuring that the contexts generated from them will be well-formed in the manner required by the logic. These constraints are represented by the typing judgements $\vdash \mathcal{B}\ \mathit{blk\ schema}$ and $\vdash \mathcal{C}\ \mathit{ctx\ schema}$, respectively, that are defined by the rules in Figure 3.4.

$$\frac{}{\Theta \vdash_{dec} \cdot \Rightarrow \Theta} \qquad \frac{\Theta \vdash_{dec} \Delta \Rightarrow \Theta' \qquad y \text{ is not assigned by } \Theta' \qquad \Theta' \vdash_{ak} A\ \mathit{type}}{\Theta \vdash_{dec} \Delta, y : A \Rightarrow \Theta' \cup \{y : (A)^-\}}$$

$$\frac{x_1, \ldots, x_n \text{ are distinct variables} \qquad \Theta_0 \cup \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\} \vdash_{dec} \Delta \Rightarrow \Theta'}{\vdash \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}\Delta\ \mathit{blk\ schema}}$$

$$\frac{}{\vdash \cdot\ \mathit{ctx\ schema}} \qquad \frac{\vdash \mathcal{C}\ \mathit{ctx\ schema} \qquad \vdash \mathcal{B}\ \mathit{blk\ schema}}{\vdash \mathcal{C}, \mathcal{B}\ \mathit{ctx\ schema}}$$

Figure 3.4: Wellformedness Judgements for Block and Context Schemas

Context Expressions	G	$::=$	$\cdot \mid \Gamma \mid G, n : A$
Formulas	F	$::=$	$\{G \vdash M : A\} \mid \top \mid \bot \mid F_1 \supset F_2 \mid F_1 \wedge F_2 \mid$
			$F_1 \vee F_2 \mid \Pi\,\Gamma : \mathcal{C}.F \mid \forall x : \alpha.F \mid \exists x : \alpha.F$

Figure 3.5: The Formulas of the Logic

We are finally in a position to describe the formulas in the logic. The syntax of these formulas is presented in Figure 3.5. The symbol Γ is used in these formulas to represent context variables. Atomic formulas, which represent LF typing judgements, have the form $\{G \vdash M : A\}$. The context in these formulas is constituted by a sequence of type associations with nominal constants, possibly preceded by a context variable. Included in the collection are the logical constants $\top$ and $\bot$ and the familiar connectives for constructing more complex formulas. Universal and existential quantification over term variables is also permitted and these are written as $\forall x : \alpha.F$ and $\exists x : \alpha.F$, respectively. Such quantification is indexed, as might be expected, by arity types. The collection also includes universal quantification over context variables that is typed by context schemas, written as $\Pi\,\Gamma : \mathcal{C}.F$. We assume the usual principle of equivalence under renaming with respect to the term and context quantifiers and apply them as needed.

$$\frac{}{\Theta; \Xi \vdash \cdot \ \mathit{context}} \qquad \frac{\Gamma \in \Xi}{\Theta; \Xi \vdash \Gamma \ \mathit{context}}$$

$$\frac{\Theta; \Xi \vdash G \ \mathit{context} \qquad n : (A)^- \in \Theta \qquad \Theta \vdash_{ak} A \ \mathit{type}}{\Theta; \Xi \vdash G, n : A \ \mathit{context}}$$

$$\frac{\Theta; \Xi \vdash G \ \mathit{context} \qquad \Theta \vdash_{ak} A \ \mathit{type} \qquad \Theta \vdash_{at} M : (A)^-}{\Theta; \Xi \vdash \{G \vdash M : A\} \ \mathit{fmla}}$$

$$\frac{}{\Theta; \Xi \vdash \top \ \mathit{fmla}} \qquad \frac{}{\Theta; \Xi \vdash \bot \ \mathit{fmla}} \qquad \frac{\Theta; \Xi \vdash F_1 \ \mathit{fmla} \qquad \Theta; \Xi \vdash F_2 \ \mathit{fmla}}{\Theta; \Xi \vdash F_1 \bullet F_2 \ \mathit{fmla}} \ \bullet \in \{\supset, \wedge, \vee\}$$

$$\frac{\vdash \mathcal{C} \ \mathit{ctx\ schema} \qquad \Theta; \Xi \cup \{\Gamma\} \vdash F \ \mathit{fmla}}{\Theta; \Xi \vdash \Pi\,\Gamma : \mathcal{C}.F \ \mathit{fmla}} \qquad \frac{\{x : \alpha\} \uplus \Theta; \Xi \vdash F \ \mathit{fmla}}{\Theta; \Xi \vdash \mathcal{Q}x : \alpha.F \ \mathit{fmla}} \ \mathcal{Q} \in \{\forall, \exists\}$$

Figure 3.6: The Wellformedness Judgement for Formulas

A formula F is determined to be well-formed or not relative to an arity context Θ and a collection of context variables Ξ. This judgement is written concretely as $\Theta; \Xi \vdash F \ \mathit{fmla}$ and the rules defining it are presented in Figure 3.6. At the top-level, formulas are expected to be closed, i.e., to not have any free term or context variables. More specifically, we expect $\mathcal{N} \cup \Theta_0; \emptyset \vdash F \ \mathit{fmla}$ to be derivable for such formulas. The analysis within the scope of term and context quantifiers augments these sets in the expected way. For context quantifiers, this analysis must also check that the annotating context schema is well-formed. An atomic formula $\{G \vdash M : A\}$ is deemed well-formed if its components G, M and A are well-formed and if M can be assigned the erased form of A as its arity type. The context expression G is well-formed if any context variable used in it is bound in the overall formula and if the types assigned to nominal constants in the explicit part of G are well-formed and such that their erased forms match the arity types of the nominal constants they are assigned to. Note that these types may use nominal constants arbitrarily; assessing whether they are used in a manner that respects dependencies is a part of the meaning of the atomic formula.

An obvious result about well-formed formulas is that the derivability of the judgement is not changed by the addition of unused bindings in either parametrizing context. This is the content of the following theorem, which is used in later proofs to ensure that we can match the context under which well-formedness has been determined with a possibly larger arity (resp. context variable) context.

Theorem 3.2. *For any F, $\Theta \subseteq \Theta'$, and $\Xi \subseteq \Xi'$, if $\Theta; \Xi \vdash F$ fmla is derivable then $\Theta'; \Xi' \vdash F$ fmla is derivable.*

Proof. This proof is completed by a straightforward induction on the well-formedness derivation, and the resulting derivation has the same structure as the given derivation. $\square$

3.2 The Interpretation of Formulas

A key component to understanding the meanings of formulas is understanding the interpretation of the quantifiers over term and context variables. These quantifiers are intended to range over closed expressions of the relevant categories. For a quantifier over a term variable, this translates concretely into closed terms of the relevant arity type. For a quantifier over a context variable, we must first explain when an LF context in which variables are represented by nominal constants satisfies a context schema.

$$\frac{}{\mathbb{N} \vdash \cdot \leadsto_{dec} \cdot \bowtie \emptyset} \qquad \frac{\mathbb{N} \vdash \Delta \leadsto_{dec} G \bowtie \theta \qquad n : (A)^- \in \mathbb{N} \qquad A[\![\theta]\!] = A'}{\mathbb{N} \vdash \Delta, y : A \leadsto_{dec} G, n : A' \bowtie \theta \cup \{\langle y, n, (A)^- \rangle\}}$$

$$\frac{\begin{array}{c}\mathbb{N} \vdash \Delta \leadsto_{dec} G' \bowtie \theta \\ G'[\![\{\langle x_i, t_i, \alpha_i \rangle \mid 1 \leq i \leq n\}]\!] = G\end{array} \qquad \{\mathbb{N} \cup \Psi \cup \Theta_0 \vdash_{at} t_i : \alpha_i \mid 1 \leq i \leq n\}}{\mathbb{N}; \Psi \vdash \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}\Delta \leadsto_{bs} G}$$

$$\frac{\mathbb{N}; \Psi \vdash \mathcal{B} \leadsto_{bs} G}{\mathbb{N}; \Psi \vdash \mathcal{C}, \mathcal{B} \leadsto^1_{cs} G} \qquad \frac{\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto^1_{cs} G}{\mathbb{N}; \Psi \vdash \mathcal{C}, \mathcal{B} \leadsto^1_{cs} G}$$

$$\frac{}{\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto_{cs} \cdot} \qquad \frac{\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto_{cs} G \qquad \mathbb{N}; \Psi \vdash \mathcal{C} \leadsto^1_{cs} G'}{\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto_{cs} G, G'}$$

Figure 3.7: Instantiating a Context Schema

We do this by describing the relation of "being an instance of" between a closed context expression G and a context schema $\mathcal{C}$. This relation is indexed by a nominal constant context $\mathbb{N}$ that is a subset of $\mathcal{N}$ and a variable context Ψ that identifies variables together with their arity types: in combination with the constants in Θ_0, these collections, circumscribe the symbols that can be used in the declarations in the context expressions.[1] The relation is written as $\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto_{cs} G$ and it is defined by the rules in Figure 3.7. This relation is defined via the repeated use of a "one-step" instantiation relation written as $\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto^1_{cs} G$; note that by G, G' we mean a context expression that is obtained by adding the bindings corresponding to G' in front of those in G. The definition of the one-step instantiation relation for context schemas uses an auxiliary judgement $\mathbb{N}; \Psi \vdash \mathcal{B} \leadsto_{bs} G$ that denotes the relation of "being an instance of" between a block schema and a context expression fragment. This relation holds when the context expression is obtained by generating a sequence of bindings for nominal constants from $\mathbb{N}$ using the body of the block schema and then instantiating the variables in the header of the block schema with terms of the right arity types. The former task is realized through the relation $\mathbb{N} \vdash \Delta \leadsto_{dec} G \bowtie \theta$

[1] In determining closed instances of context schemas, $\mathbb{N}$ will be $\mathcal{N}$ and Ψ will be the empty set. The more general form for this relation, which includes a parameterization by these sets, will be useful in later sections.

that holds between a block of declarations Δ, a context expression G that is obtained by replacing the variables assigned in Δ with suitable nominal constants, and a substitution θ that corresponds to this replacement. We assume here and elsewhere that the application of a hereditary substitution to a sequence of declarations corresponds to its application to the type in each assignment.

Theorem 3.3. *Let $\mathcal{C}$ and G be a context schema and a context expression such that both $\vdash \mathcal{C}$ ctx schema and $\mathbb{N}; \Psi \vdash \mathcal{C} \leadsto_{cs} G$ are derivable. Then for any arity context Θ such that $\mathbb{N} \cup \Psi \cup \Theta_0 \subseteq \Theta$, it is the case that $\Theta; \emptyset \vdash G$ context has a derivation.*

Proof. We first show that for any block declaration Δ and any arity context Θ such that $\mathbb{N} \subseteq \Theta$, if $\Theta \vdash_{dec} \Delta \Rightarrow \Theta'$ and $\mathbb{N} \vdash \Delta \leadsto_{dec} G' \bowtie \theta'$ are derivable for some Θ' and θ, then (a) θ' is type preserving with respect to Θ, (b) Θ' is $ctx(\theta) \uplus \Theta$, and (c) each binding in G' is of the form $n : A$ where $n : (A)^{-} \in \Theta$ and $\Theta \vdash_{ak} A$ *type* has a derivation. This claim is proved by induction on the derivation of $\Theta \vdash_{dec} \Delta \Rightarrow \Theta'$; properties (a) and (b) are included in the claim because they are useful together with Theorem 3.1 in showing property (c) in the induction step. Next we show, through an easy inductive argument, that if $\Theta_0 \cup \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\} \vdash_{dec} \Delta \Rightarrow \Theta'$ has a derivation and Θ is such that $\mathbb{N} \cup \Psi \cup \Theta_0 \subseteq \Theta$, then, for some Θ'', it is the case that $\{x_1 : \alpha_1, \ldots, x_n : \alpha_n\} \uplus \Theta \vdash_{dec} \Delta \Rightarrow \Theta''$ has a derivation. Using Theorem 3.1 with these two observations, we can show easily that if $\vdash \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}\Delta$ *blk schema* and $\mathbb{N}; \Psi \vdash \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}\Delta \leadsto_{bs} G$ have derivations then for each binding of the form $n : A$ in G it is the case that $n : (A)^{-} \in \Theta$ and $\Theta \vdash_{ak} A$ *type*. The theorem follows easily from this observation. □

In defining validity for formulas, we need to consider substitutions for context and term variables. For context variables this will correspond to the naive replacement of the free occurrences of the variables by given context expressions. We write $F[G_1/\Gamma_1, \ldots, G_1/\Gamma_n]$ to denote the result of the replacement, for $1 \leq i \leq n$, of Γ_i by G_i. For term variables, the replacement must also ensure the transformation of the resulting expression to normal form. Towards this end, we adapt hereditary substitution to formulas. The application of this substitution simply distributes over quantifiers and logical symbols, respecting the scopes of quantifiers through the necessary renaming. The application to the atomic formula $\{G \vdash M : A\}$ also distributes to the component parts. We have already discussed the application to terms and types. The application to context expressions leaves context variables unaffected and simply distributes to the types in the explicit bindings. In particular, no check is made of the possibility of inadvertent capture. In this respect, this application is unlike that to LF contexts that is defined in Figure 2.4.

Theorem 3.4. *Let Θ be an arity context and let Ξ be a collection of context variables.*

1. *If θ is a term variable substitution that is arity type preserving with respect to Θ and F is a formula such that there is a derivation for $ctx(\theta) \uplus \Theta; \Xi \vdash F$ fmla, then there is a unique formula F' such that $F[\![\theta]\!] = F'$ has a derivation. Moreover, for this F' it is the case that $\Theta; \Xi \vdash F'$ fmla is derivable.*

2. *If $\sigma = \{G_1/\Gamma_1, \ldots, G_n/\Gamma_n\}$ is a context variable substitution which is such that all judgements in the collection $\{\Theta; \Xi \setminus \{\Gamma_1, \ldots, \Gamma_n\} \vdash G_i$ context $\mid 1 \leq i \leq n\}$ are derivable and F is a formula such that there is a derivation for $\Theta; \Xi \vdash F$ fmla, then there is a derivation for $\Theta; \Xi \setminus \{\Gamma_1, \ldots, \Gamma_n\} \vdash F[\sigma]$ fmla.*

Proof. The first clause is easily provable by induction on $ctx(\theta) \uplus \Theta; \Xi \vdash F$ *fmla*, using Theorems 2.1 and 2.4 in the atomic case to ensure the appropriate arity typing judgements will be derivable under the substitution θ.

The second clause is by induction on the structure of $\Theta; \Xi \vdash F$ *fmla*, using the assumption derivations that $\Theta; \Xi \setminus \{\Gamma_1, \ldots, \Gamma_n\} \vdash G_i$ *context* is derivable to ensure the result of the substitution on a context variable is well-formed in the atomic case. □

Following the notation introduced after Theorem 2.4, if F and θ are a formula and a substitution that together satisfy the requirements of the first part of the theorem, we will write $F[\![\theta]\!]$ to denote the F' for which $F[\![\theta]\!] = F'$ is derivable. Note that a term variable substitution may introduce new nominal constants. We will write $supp(\theta)$ to denote the collection of such constants that appear in $rng(\theta)$. A similar observation holds for context variable substitutions: if σ is the substitution $\{G_1/\Gamma_1, \ldots, G_n/\Gamma_n\}$, we will write $supp(\sigma)$ to denote the collection of nominal constants that appear in $G_1, \ldots, G_n$.

As discussed previously, a closed atomic formula of the form $\{G \vdash M : A\}$ is intended to encode an LF judgement of the form $\Gamma \vdash_\Sigma M \Leftarrow A$. In this encoding, nominal constants that appear in terms represent free variables for which bindings appear in the context in LF judgements. To substantiate this interpretation, the rules *CANON_KIND_PI*, *CANON_FAM_PI* and *CANON_TERM_LAM* must introduce fresh nominal constants into contexts in typing derivations and they must replace bound variables appearing in terms and types with these constants. We use this interpretation to define validity for closed atomic formulas with one further qualification: unlike in the LF judgement, for the atomic formula we must also ascertain the wellformedness of the context and the type. This notion of validity is then extended to all closed formulas by recursion on formula structure as we describe below.

Definition 3.2. Let F be a formula such that $\mathcal{N} \cup \Theta_0; \emptyset \vdash F\ \mathit{fmla}$ is derivable.

- If F is $\top$ it is valid and if it is $\bot$ it is not valid.
- If F is $\{G \vdash M : A\}$, it is valid exactly when all of $\vdash_\Sigma G\ \mathtt{ctx}$, $G \vdash_\Sigma A\ \mathtt{type}$, and $G \vdash_\Sigma M \Leftarrow A$ are derivable in LF, under the interpretation of nominal constants as variables bound in a context and with the modification of the rules *CANON_KIND_PI*, *CANON_FAM_PI* and *CANON_TERM_LAM* to introduce fresh nominal constants into contexts and to instantiate the relevant bound variables in kinds, types and terms with these constants.
- If F is $F_1 \supset F_2$, it is valid if F_2 is valid in the case that F_1 is valid.
- If F is $F_1 \wedge F_2$, it is valid if both F_1 and F_2 are valid.
- If F is $F_1 \vee F_2$, it is valid if either F_1 or F_2 is valid.

- If F is $\Pi\,\Gamma : \mathcal{C}.F$, it is valid if $F[G/\Gamma]$ is valid for every G such that $\mathcal{N};\emptyset \vdash \mathcal{C} \rightsquigarrow_{cs} G$ is derivable.
- If F is $\forall x : \alpha.F$, it is valid if $F[\![\{\langle x, M, \alpha\rangle\}]\!]$ is valid for every term M such that $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ is derivable.
- If F is $\exists x : \alpha.F$, it is valid if $F[\![\{\langle x, M, \alpha\rangle\}]\!]$ is valid for some term M such that $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ is derivable.

Theorem 3.4 guarantees the coherence of this definition.

3.3 Understanding the Notion of Validity

In the examples we consider below, we assume an instantiation of the logic based on the signature presented in Section 2.2. Obviously, any LF typing judgement based on that signature is expressable in the logic. Moreover, the corresponding formula will be valid exactly when the typing judgement is derivable in LF. Thus, the formulas $\{\cdot \vdash \mathit{empty} : \mathit{tm}\}$, $\{\cdot \vdash (\mathit{lam}\ \mathit{unit}\ (\lambda x.\, x)) : \mathit{tm}\}$ and $\{n : \mathit{tm} \vdash n : \mathit{tm}\}$ are all valid. Similarly, the formulas $\exists d : o.\, \{\cdot \vdash d : (\mathit{of}\ \mathit{empty}\ \mathit{unit})\}$ and

$$\exists d : o.\, \{\cdot \vdash d : \mathit{of}\ (\mathit{lam}\ \mathit{unit}\ (\lambda x.\, x))\ (\mathit{arr}\ \mathit{unit}\ \mathit{unit})\}$$

are valid but the formula $\exists d : o.\, \{\cdot \vdash d : \mathit{of}\ (\mathit{lam}\ \mathit{unit}\ (\lambda x.\, x))\ \mathit{unit}\}$ is not. Note that the arity type associated with the quantified variable in each of these formulas provides only a rough constraint on the instantiation needed to verify the validity of the formula; to do this, the instance must also satisfy LF typeability requirements represented by formula that appears within the scope of the quantifier.

Wellformedness conditions for formulas ensure only that the terms appearing within formulas satisfy canonicity requirements, i.e. that these terms are in β-normal form and that variables and constants are applied to as many arguments as they can take. Arity typing does not distinguish between terms in different expression categories. For example, the formula

$$\exists d : o. \{\cdot \vdash d : \mathit{of}\ (\mathit{lam\ empty}\ (\lambda x. x))\ (\mathit{arr\ unit\ unit})\}$$

is well-formed but not valid. An alternative design choice, with equivalent consequences from the perspective of the valid properties that can be expressed in the logic, might have been to let the fact that *lam* is ill-applied to *empty* impact on the wellformedness of the formula. The wellformedness conditions do not also enforce a distinctness requirement for bindings in a context. Thus, the formula $\{n : \mathit{tm}, n : \mathit{tp} \vdash \mathit{empty} : \mathit{tm}\}$ is well-formed. However, it is not valid because $\vdash_\Sigma n : \mathit{tm}, n : \mathit{tp}\ \mathtt{ctx}$ is not derivable in LF under the described interpretation for nominal constants. An implication of these observations is that a naive form of weakening does not hold with respect to the encoding of LF derivability in the logic; additional conditions similar to this described in Theorem 2.8 must be verified for this principle to apply.

To provide a more substantive example of the kinds of properties that can be expressed in the logic, let us consider the formal statement of the uniqueness of type assignment for simply typed λ-calculus terms. As noted in Section 2.2, this property is best described in a form that considers typing expressions in contexts that have a particular kind of structure. That structure can be formalized in the logic by a context schema comprising the single block

$$\{t : o\} x : \mathit{tm}, y : \mathit{of}\ x\ t.$$

Let us denote this context schema by ctx. Observe that any context that instantiates this schema will not provide a variable that can be used to construct an atomic term of type tp. Thus, the strengthening property for expressions representing types that is expressed by the formula

$$\Pi\,\Gamma : \mathit{ctx}. \forall t : o. \{\Gamma \vdash t : \mathit{tp}\} \supset \{\cdot \vdash t : \mathit{tp}\}.$$

should hold. We can in fact easily show this formula to be valid by using Theorem 2.12 and an induction on the height of the derivation for $\{G \vdash t : \mathit{tp}\}$ for a closed term t and a closed instance G of ctx. Using the validity of this formula, we can also easily argue that the following formula that expresses a strengthening property pertaining to the equality of types is also valid:

$$\Pi\,\Gamma : ctx.\forall d : o.\forall t_1 : o.\forall t_2 : o.\,\{\Gamma \vdash d : eq\ t_1\ t_2\} \supset \{\cdot \vdash d : eq\ t_1\ t_2\}.$$

Then the uniqueness of typing property can be expressed through the following formula:

$$\begin{array}{l}\Pi\,\Gamma : ctx.\forall e : o.\forall t_1 : o.\forall t_2 : o.\forall d_1 : o.\forall d_2 : o.\\ \qquad \{\Gamma \vdash d_1 : of\ e\ t_1\} \supset \{\Gamma \vdash d_2 : of\ e\ t_2\} \supset \exists d_3 : o.\,\{\cdot \vdash d_3 : eq\ t_1\ t_2\}\end{array}$$

This formula can be seen to be valid using the strengthening property just described if we can establish the validity of the formula

$$\begin{array}{l}\Pi\,\Gamma : ctx.\forall e : o.\forall t_1 : o.\forall t_2 : o.\forall d_1 : o.\forall d_2 : o.\\ \qquad \{\Gamma \vdash d_1 : of\ e\ t_1\} \supset \{\Gamma \vdash d_2 : of\ e\ t_2\} \supset \exists d_3 : o.\,\{\Gamma \vdash d_3 : eq\ t_1\ t_2\}.\end{array}$$

To show this, it suffices to argue that, for a closed context expression G that instantiates the schema ctx and for closed expressions d_1, d_2, e, t_1, and t_2, if $\{G \vdash d_1 : of\ e\ t_1\}$ and $\{G \vdash d_2 : of\ e\ t_2\}$ are valid, then there must be a closed expression d_3 such that $\{G \vdash d_3 : eq\ t_1\ t_2\}$ is also valid. Such an argument can be constructed by induction on the height of the LF derivation of $G \vdash_\Sigma d_1 \Leftarrow of\ e\ t_1$, which we analyze using Theorem 2.12 in the manner discussed earlier. There are essentially four cases to consider, corresponding to whether the head symbol of d_1 is *of_empty*, *of_app*, *of_lam*, or a nominal constant that is assigned the type $(of\ n\ t_1)$ in G where n is also a nominal constant that is bound in G. In the last case, we use the fact that the validity of $\{G \vdash d_1 : of\ e\ t_1\}$ implies that $\vdash_\Sigma G$ `ctx` is derivable to conclude the uniqueness of n and, hence, of the typing. The argument when d_1 is *of_empty* has an obvious form. The argument when d_1 has *of_app* or *of_lam* as its head symbol will invoke the induction hypothesis. In the case where the head symbol is *of_lam*, we will need to consider a shorter derivation of a typing judgement in which the context has been enhanced. However, we will be able to use the induction hypothesis by observing that the enhancements to the context conform to the constraints imposed by the context schema. Note that the form of d_1 also constrains the form of e in all the cases, a fact that is used implicitly in the analysis outlined.

3.4 Nominal Constants and Invariance Under Permutations

As noted in Chapter 2, the particular choices for bound variable names in the kinds, types and terms that comprise LF expressions are considered irrelevant. This understanding is built in concretely through the notion of α-conversion that renders equivalent expressions that differ only in the names used for such variables. Typing derivations transform expressions with bound variables into ones where variables are ostensibly free but in fact bound implicitly in the associated contexts. The lack of importance of name choices is reflected in this case in an invariance in the validity of typing judgements under a suitable renaming of variables appearing in the judgements. In a situation where context variables are represented by nominal constants, this property has a simple expression in the form of an invariance of formula validity under permutations of nominal constants. We will need this property in later chapters and so we present it formally below.

We begin with a definition of the notions of permutations of nominal constants and their applications to expressions.

Definition 3.3 (Permutation)**.** A permutation of the nominal constants is an arity type preserving bijection from $\mathcal{N}$ to $\mathcal{N}$ that differs from the identity map at only a finite number of constants. The permutation that maps $n_1, \ldots, n_m$ to $n'_1, \ldots, n'_m$, respectively, and is the identity everywhere else is written as $\{n'_1/n_1, \ldots, n'_m/n_m\}$. The support of a permutation $\pi = \{n'_1/n_1, \ldots, n'_m/n_m\}$, denoted by $\mathrm{supp}\,(\pi)$, is the collection of nominal constants $\{n_1, \ldots, n_m\} \cup \{n'_1, \ldots, n'_m\}$. Every permutation π has an obvious inverse that is written as π^{-1}.

Definition 3.4 (Permutation Application)**.** The application of a permutation π to an expression E of a variety of kinds is described below and is denoted in all cases by $\pi.E$. If E is a term, type, or kind then the application consists of replacing each nominal constant n that appears in E with $\pi(n)$. If E is a context then the application of π to E replaces each explicit binding $n : A$ in E with $\pi(n) : \pi.A$. If E is an LF judgement $\mathcal{J}$ then the permutation is applied to each component of the judgement in the way described above.

If E is a formula then the permutation is applied to its component parts. The application of π to a term variable substitution $\{\langle x_1, M_1, \alpha_1\rangle, \ldots, \langle x_n, M_n, \alpha_n\rangle\}$ yields the substitution $\{\langle x_1, \pi.M_1, \alpha_1\rangle, \ldots, \langle x_n, \pi.M_n, \alpha_n\rangle\}$. The application of π to a context variable substitution $\{G_1/\Gamma_1, \ldots, G_n/\Gamma_n\}$ yields $\{\pi.G_1/\Gamma_1, \ldots, \pi.G_n/\Gamma_n\}$.

The following theorem expresses the property of interest concerning LF judgements cast in the form relevant to the logic.

Theorem 3.5. *Let LF judgements and derivations be recast in the form discussed earlier in this section: variables that are bound in a context are represented by nominal constants and the rules CANON_KIND_PI, CANON_FAM_PI and CANON_TERM_LAM introduce fresh nominal constants into contexts and replace variables in kinds, types and terms with these constants. In this context, let $\mathcal{J}$ be an LF judgement which has a derivation. Then for any permutation π, $\pi.\mathcal{J}$ is derivable. Moreover, the structure of this derivation is the same as that for $\mathcal{J}$.*

Proof. This proof is by induction on the derivation for $\mathcal{J}$. Perhaps the only observation worthy of note is that the freshness of nominal constants used in *CANON_KIND_PI*, *CANON_FAM_PI*, and *CANON_TERM_LAM* rules is preserved under permutations of nominal constants. $\square$

The above observation underlies the main theorem of this section.

Theorem 3.6. *Let F be a closed formula and let π be a permutation. Then F is valid if and only if $\pi.F$ is valid.*

Proof. Noting that π^{-1} is also a permutation and that $\pi^{-1}.\pi.F$ is F, it suffices to prove the claim in only one direction. We do this by induction on the structure of F.

The desired result follows easily from Theorem 3.5 and the relationship of validity to LF derivability when F is atomic. The cases where F is $\top$ or $\bot$ are trivial and the ones in which F is $F_1 \supset F_2$, $F_1 \wedge F_2$ or $F_1 \vee F_2$ are easily argued with recourse to the induction hypothesis and by noting that the permutation distributes to the component formulas.

In the case where F is $\Pi\,\Gamma : \mathcal{C}.F'$, we first note that if $\mathcal{N}; \emptyset \vdash \mathcal{C} \leadsto_{cs} G$ has a derivation then $\mathcal{N}; \emptyset \vdash \mathcal{C} \leadsto_{cs} \pi^{-1}.G$ must also have one. From this and the validity of F it follows that $F'[\pi^{-1}.G/\Gamma]$ must be valid. Moreover, $F'[\pi^{-1}.G/\Gamma]$ has the same structural complexity as F'. Hence, by the induction hypothesis, $\pi.(F'[\pi^{-1}.G/\Gamma])$ is valid. Noting that this formula is the same as $(\pi.F')[G/\Gamma]$ and that $\Pi\,\Gamma : \mathcal{C}.\pi.F'$ is identical to $\pi.(\Pi\,\Gamma : \mathcal{C}.F')$, the validity of $\pi.F$ easily follows.

Suppose that F has the form $\forall x : \alpha.F'$. We observe here that if $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ has a derivation then $\mathcal{N} \cup \Theta_0 \vdash_{at} \pi^{-1}.M : \alpha$ has one too and that $\pi.(F'[\![\{\langle x, \pi^{-1}.M, \alpha\rangle\}]\!])$ is the same formula as $(\pi.F')[\![\{\langle x, M, \alpha\rangle\}]\!]$. Using the definition of validity, the induction hypothesis and the fact that permutation distributes to the component formula together with the above observations, we may easily conclude that $\pi.F$ is valid.

Finally, suppose that F is of the form $\exists x : \alpha.F'$. Here we note that if $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ has a derivation then $\mathcal{N} \cup \Theta_0 \vdash_{at} \pi.M : \alpha$ has one too and that $\pi.(F'[\![\{\langle x, M, \alpha\rangle\}]\!])$ is the same formula as $(\pi.F')[\![\{\langle x, \pi.M, \alpha\rangle\}]\!]$. Using the definition of validity and the induction hypothesis, it is now easy to conclude that $\pi.F$ must be valid. $\square$

Chapter 4

A Proof System for Constructing Arguments of Validity

We have presented a logic in Chapter 3 that can be used to describe properties of LF specifications. We have also shown there how we can argue informally about the validity of formulas that encapsulate such properties. Our goal now is to develop a formal mechanism for constructing arguments of validity. Towards this end, we describe in this chapter a proof system that complements our logic. This proof system is oriented around sequents that represent assumption and conclusion formulas augmented with devices that capture additional aspects of states that arise in the process of reasoning. The syntax for sequents is more liberal than is meaningful at the outset, and this is rectified by imposing wellformedness requirements on them. We associate a semantics with sequents that is consistent with their intended use in enabling reasoning about the validity of formulas. We then present a collection of proof rules that can be used to derive sequents. These rules belong to two broad categories. The first category comprises rules that embody logical aspects such as the meanings of sequents and of the logical symbols that appear in formulas. A key aspect of our logic is that its atomic formulas represent the notion of derivability in LF that is also open to analysis. The second category of proof rules builds in capabilities for such analysis. To be coherent, our proof rules must preserve the wellformedness property of sequents and our first endeavor concerning their presentation is to show that they indeed satisfy this property. At a more substantive level, the proof rules must support a reasoning process that is both sound and effective. The focus in this chapter is on ensuring soundness. We do this by demonstrating for each proposed rule that the conclusion sequent must be valid if its premise sequents are. The demonstration of effectiveness for the proof system will be the subject of later chapters.

The first section below presents the sequents underlying the proof system and identifies a semantics that enables their use in validity arguments and that also undergirds the demonstrations of soundness of proof rules. The remaining sections in the chapter develop the collection of proof rules. In Section 4.2, we present the "core" rules, i.e., the rules that encapsulate the meanings of the logical symbols and also certain aspects of sequents. We then turn to the rules that internalize aspects of LF derivability that permeate the logic through the interpretation of the atomic formulas. Section 4.3 develops rules for analyzing atomic formulas. An important component of these rules is the interpretation of typing judgements involving atomic types via the particular LF specification that parameterizes the logic: this interpretation leads, in particular, to a case analysis rule for such atomic formulas that appear as assumptions in a sequent. Another important component of informal reasoning that needs to be supported by the proof system is that which is based on an induction on the height of an LF derivation. To support this ability, we introduce an induction rule in Section 4.4 that is inspired by the annotation based scheme used in Abella [BCG$^+$14, Gac09b]. In the final section of the chapter, we introduce proof rules that encode meta-theorems concerning LF derivability that often find use in reasoning about LF specifications.

4.1 The Structure of Sequents

A sequent in our proof system is characterized by a collection of assumption formulas and a conclusion or goal formula. The formulas may contain free term and context variables that are to be interpreted as being implicitly universally quantified over the sequent and, therefore, its proof. We find it useful also to identify with the sequent a collection of nominal constants that circumscribes the ones that appear in its formulas.

The nominal constants and term variables that appear in the sequent have arity types associated with them. Context variables are also typed and their types are, in spirit, based on context schemas. However, subproofs may require a partial elaboration of a context variable and the types associated with such variables accommodates this possibility. More

$$\frac{}{\mathbb{N};\Psi \vdash \mathcal{C}[\cdot] \text{ ctx-ty}} \qquad \frac{\mathbb{N};\Psi \vdash \mathcal{C} \leadsto^1_{cs} G \qquad \mathbb{N};\Psi \vdash \mathcal{C}[\mathcal{G}] \text{ ctx-ty}}{\mathbb{N};\Psi \vdash \mathcal{C}[\mathcal{G}; G] \text{ ctx-ty}}$$

$$\frac{}{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\cdot] \leadsto_{csty} \cdot} \qquad \frac{\Gamma\!\uparrow\!\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi \quad (\mathcal{N} \setminus \mathbb{N}_\Gamma) \subseteq \mathbb{N}}{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\cdot] \leadsto_{csty} \Gamma}$$

$$\frac{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G \qquad \mathbb{N};\Psi \vdash \mathcal{C} \leadsto^1_{cs} G'}{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G, G'} \qquad \frac{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G}{\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\mathcal{G}; G'] \leadsto_{csty} G, G'}$$

Figure 4.1: Well-Formed Context Variable Types and their Instantiations

specifically, these types have the form $\mathcal{C}[G_1; \ldots; G_n]$ where $\mathcal{C}$ is a context schema and $G_1, \ldots, G_n$ are context expressions. Such a type is intended to represent the collection of context expressions obtained by interspersing $G_1, \ldots, G_n$ with instantiations of the context schema $\mathcal{C}$ and possibly prefixed by a context variable of suitable type that represents a yet to be elaborated sequence of declarations. Additionally, context variables are annotated with a collection of nominal constants that express the constraint that the elaborations of these variables must not use names in these collections; the ability to express such constraints is an essential part of the mechanism for analysing typing judgements involving abstractions as we will see later in this section.

The ideas pertaining to context variable typing are made precise through the following definition.

Definition 4.1 (Context Variable Types and their Instances)**.** A *context variable type* is a expression of the form $\mathcal{C}[\mathcal{G}]$ where $\mathcal{C}$ is a context schema such that $\vdash \mathcal{C}$ *ctx schema* is derivable and $\mathcal{G}$ represents a sequence of *context blocks* given as follows:

$$\mathcal{G} ::= \cdot \mid \mathcal{G}; n_1 : A_1, \ldots, n_k : A_k.$$

Such a type is said to be well-formed with respect to a nominal constant set $\mathbb{N} \subseteq \mathcal{N}$ and a term variable context Ψ that associates arity types with term variables if it is the case that the relation $\mathbb{N};\Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty that is defined by the rules in Figure 4.1 holds. A *context variable context* is a collection of associations of sets of nominal constants and context variable types with context variables, each written in the form $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}]$. Given a context variable context Ξ, we write Ξ^- for the set $\{\Gamma \mid \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi\}$, i.e., the collection of context variables assigned types by Ξ. A context expression G is said to be an instance of a context type $\mathcal{C}[\mathcal{G}]$ relative to $\mathbb{N}$, Ψ and the context variable context Ξ if the relation $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \rightsquigarrow_{csty} G$, that is also defined in Figure 4.1, holds.

The following theorem, whose proof is based on an obvious induction, shows that an instance of a well-formed context variable type is a well-formed context relative to the relevant arity context and context variable collection.

Theorem 4.1. *If $\mathbb{N}; \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty and $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \rightsquigarrow_{csty} G$ have derivations then so does $\mathbb{N} \cup \Psi; \Xi \vdash G$ context.*

In using the theorem above and in other similar situations, we will often need to adjust the contexts that parameterize the relevant wellformedness judgements. The theorem below, whose proof is also based on a straightforward induction, provides the basis for such adjustments.

Theorem 4.2. *Let $\mathbb{N} \subseteq \mathbb{N}'$, $\Psi \subseteq \Psi'$, and $\Xi \subseteq \Xi'$.*

1. *If $\mathbb{N}; \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty is derivable, then $\mathbb{N}'; \Psi' \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty is derivable.*

2. *If $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \rightsquigarrow_{csty} G$ is derivable, then $\mathbb{N}'; \Psi'; \Xi' \vdash \mathcal{C}[\mathcal{G}] \rightsquigarrow_{csty} G$ is derivable.*

The wellformedness judgements in Figure 4.1 are preserved under meaningful substitutions as the theorem below explicates.

Theorem 4.3. *Let θ be an arity type preserving substitution with respect to $\mathcal{N} \cup \Theta_0 \cup \Psi$ and let $\mathbb{N}; ctx(\theta) \uplus \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty have a derivation. Then*

1. *there must be a derivation for $\mathbb{N} \cup supp(\theta); \Psi \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!]]$ ctx-ty, and*

2. *if $\mathbb{N}; ctx(\theta) \uplus \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \rightsquigarrow_{csty} G$ also has a derivation, there must be a derivation for $\mathbb{N} \cup supp(\theta); \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!]] \rightsquigarrow_{csty} G[\![\theta]\!]$.*

Proof. Since θ is arity type preserving with respect to $\mathcal{N} \cup \Theta_0 \cup \Psi$, using Theorem 2.4 we see that if there is a derivation for $\mathbb{N} \cup (ctx(\theta) \uplus \Psi) \cup \Theta_0 \vdash_{at} t : \alpha$ then $t[\![\theta]\!]$ is well-defined and $(\mathbb{N} \cup supp(\theta) \cup \Psi \cup \Theta_0 \vdash_{at} t[\![\theta]\!] : \alpha$ has a derivation. An induction on the derivation of $\mathbb{N}; ctx(\theta) \uplus \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty using these observations allows us to confirm the first part of the theorem. Further, let there be a derivation for $\mathbb{N}; ctx(\theta) \uplus \Psi \vdash \mathcal{C} \rightsquigarrow^1_{cs} G'$. By an induction on this derivation using the facts observed earlier, it can be concluded that there must be a derivation for $\mathbb{N} \cup supp(\theta); \Psi \vdash \mathcal{C} \rightsquigarrow^1_{cs} G'[\![\theta]\!]$. The second part of the theorem follows by another obvious induction from this. □

Definition 4.2 (Sequents)**.** A sequent, written as $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$, is a judgement that relates a finite subset $\mathbb{N}$ of $\mathcal{N}$, a finite set Ψ of arity type assignments to term variables, a context variable context Ξ, a finite set Ω of *assumption formulas* and a conclusion or goal formula F. The sequent is well-formed if (a) for each type association $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}]$ in Ξ it is the case that $\mathbb{N} \setminus \mathbb{N}_\Gamma; \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty is derivable and (b) for each formula F' in $\{F\} \cup \Omega$ the judgement $\mathbb{N} \cup \Psi \cup \Theta_0; \Xi^- \vdash F'$ *fmla* is derivable. Given a well-formed sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$, we refer to $\mathbb{N}$ as its support set, to Ψ as its eigenvariable context and Ξ as its context variable context. We use a comma to denote set union in representing sequents, writing $\Omega, F_1, \ldots, F_n$ to denote the set $\Omega \cup \{F_1, \ldots, F_n\}$.

We will need to consider substitutions for term variables in sequents. We will require legitimate substitutions to not use the nominal constants in the support set of the sequent; this restriction will be part of a mechanism for controlling dependencies in context declarations. We will further require substitutions to satisfy arity typing constraints for their applications to be well-defined. These considerations are formalized below in a notion of compatibility between substitutions and sequents.

Definition 4.3 (Term Substitutions Compatible with Sequents)**.** A pair $\langle \theta, \Psi' \rangle$ consisting of a term variable substitution and an arity context assigning types to variables is said to be substitution compatible with a well-formed sequent $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ if

1. θ is arity type preserving with respect to the context $\mathcal{N} \cup \Theta_0 \cup \Psi'$,

2. $supp(\theta) \cap \mathbb{N} = \emptyset$, and

3. for any variable x, if $x : \alpha \in \Psi$ and $x : \alpha' \in ctx(\theta) \uplus \Psi'$, then $\alpha = \alpha'$.

The application of a substitution may introduce new nominal constants into a sequent. When this happens, substitutions for the eigenvariables in the resulting sequent must be permitted to contain these constants. We use the technique of raising to realize this requirement [Mil92]. The following definition is useful in formalizing this idea.

Definition 4.4 (Raising a Context over Nominal Constants)**.** Let Ψ be a set of the form $\{x_1 : \alpha_1, \ldots, x_m : \alpha_m\}$ that associates arity types with a finite collection of variables, let $n_1, \ldots, n_k$ be a listing of the elements of a finite collection of the nominal constants $\mathbb{N}$, and let $\beta_1, \ldots, \beta_k$ be the arity types associated by $\mathcal{N}$ with these constants. Then a version of Ψ raised over $\mathbb{N}$ is a set $\{y_1 : \gamma_1, \ldots, y_m : \gamma_m\}$ where, for $1 \leq i \leq m$, y_i is a distinct variable that is also different from the variables in $\{x_1, \ldots, x_m\}$ and γ_i is $\beta_1 \to \cdots \to \beta_k \to \alpha_i$. Further, the raising substitution associated with this version is the set $\{\langle x_i, (y_i\ n_1\ \ldots\ n_k), \alpha_i\rangle \mid 1 \leq i \leq m\}$.

The basis for using raising in the manner described is the content of the following theorem. We say here and elsewhere that an arity context Θ is compatible with $\mathcal{N}$ if the types that Θ assigns to nominal constants are identical to their assignments in $\mathcal{N}$.

Theorem 4.4. *Let θ be a substitution that is arity type preserving with respect to an arity context Θ that is compatible with $\mathcal{N}$. Further, let Ψ be a version of $\mathrm{ctx}(\theta)$ raised over some listing of a collection $\mathbb{N}$ of nominal constants and let θ_r be the associated raising substitution. Then there is a substitution θ' with $supp(\theta') = supp(\theta) \setminus \mathbb{N}$ and $ctx(\theta') = \Psi$ that is arity type preserving with respect to Θ and such that for any E for which $ctx(\theta) \uplus \Theta \vdash_{ak} E$ type or, for some arity type α, $ctx(\theta) \uplus \Theta \vdash_{at} E : \alpha$ has a derivation, it is the case that $E[\![\theta_r]\!][\![\theta']\!] = E[\![\theta]\!]$.*

Proof. Each of the substitutions involved in the expression $E[\![\theta_r]\!][\![\theta']\!] = E[\![\theta]\!]$ will have a result under the conditions described, thereby justifying the use of the notation introduced after Theorem 2.4. Now, let $n_1, \ldots, n_k$ be the listing of the constants in $\mathbb{N}$ in the raising substitution, let $\alpha_1, \ldots, \alpha_k$ be the respective types of these constants, let $\langle x, t, \alpha \rangle$ be a tuple in θ and let $\langle x, (y\ n_1\ \cdots\ n_k), \alpha \rangle$ be the tuple corresponding to x in θ_r. Let $x_1, \ldots, x_k$ be a listing of distinct variables that do not appear in t and let t' be the result of replacing n_i by x_i in t, for $1 \leq i \leq k$. We construct θ' by including in it the substitution $\langle y, \lambda x_1. \ldots \lambda x_k. t', \alpha_1 \to \cdots \to \alpha_k \to \alpha \rangle$ for each case of the kind considered. It is easy to see that $supp(\theta') = supp(\theta) \setminus \mathbb{N}$ and that θ_r and θ' are arity type compatible with respect to $\Theta \cup \mathcal{N}$. The remaining part of the theorem follows from noting that $\theta = \theta' \circ \theta_r$ and using Theorem 2.5. □

The following definition formalizes the application of a term substitution to a well-formed sequent when the conditions of substitution compatibility are met. We assume here and elsewhere that the application of a substitution to a set of formulas distributes to each member of the set, its application to a context variable context distributes to each context variable type in the context and its application to a context variable type $\mathcal{C}[\mathcal{G}]$ distributes to each context block in $\mathcal{G}$.

Definition 4.5 (Applying a Term Substitution to a Sequent)**.** Let $\mathcal{S}$ be the well-formed sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ and $\langle \theta, \Psi' \rangle$ be substitution compatible with $\mathcal{S}$. Further, let Ψ'' be a version of $(\Psi \setminus ctx(\theta)) \cup \Psi'$ raised over $supp(\theta)$ and let θ_r be the corresponding raising substitution. Then the application of θ to $\mathcal{S}$ relative to Ψ', denoted by $\mathcal{S}[\![\theta]\!]_{\Psi'}$, is the sequent $\mathbb{N} \cup supp(\theta); \Psi''; \Xi[\![\theta]\!][\![\theta_r]\!]; \Omega[\![\theta]\!][\![\theta_r]\!] \longrightarrow F[\![\theta]\!][\![\theta_r]\!]$.

The definition and notation above are obviously ambiguous since they depend on the particular choices of Ψ'' and θ_r. We shall mean $\mathcal{S}[\![\theta]\!]_{\Psi'}$ to denote any one of the sequents so determined, referring to Ψ'' and θ_r as the raised context and the raising substitution associated with the application of the substitution where disambiguation is needed. Note also that the definition assumes that the application of the substitutions θ and θ_r to the relevant context variable types and formulas is well-defined. We show this to be the case in the theorem below.

Theorem 4.5. *Let $\langle \theta, \Psi' \rangle$ be substitution compatible with a well-formed sequent $\mathcal{S}$. Then $\mathcal{S}[\![\theta]\!]_{\Psi'}$ is well-defined and is a well-formed sequent.*

Proof. Let the sequent $\mathcal{S}$ be $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$. Following Definition 4.5, let Ψ'' be the context $(\Psi \setminus ctx(\theta)) \cup \Psi'$ raised over $supp(\theta)$ and θ_r the corresponding raising substitution. We must then show the following:

1. for each $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}[\![\theta]\!][\![\theta_r]\!]] \in \Xi[\![\theta]\!][\![\theta_r]\!]$, it is the case that $\mathbb{N}_\Gamma \subseteq (\mathbb{N} \cup supp(\theta))$ and $(\mathbb{N} \cup supp(\theta)) \setminus \mathbb{N}_\Gamma; \Psi'' \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!][\![\theta_r]\!]]$ ctx-ty has a derivation, and

2. for each formula $F'[\![\theta]\!][\![\theta_r]\!] \in \{F[\![\theta]\!][\![\theta_r]\!]\} \cup \Omega[\![\theta]\!][\![\theta_r]\!]$, it is the case that the judgement $\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup \Psi''; \Xi[\![\theta]\!][\![\theta_r]\!]^- \vdash F'[\![\theta]\!][\![\theta_r]\!]$ *fmla* has a derivation.

In showing these two requirements we will also ensure that the relevant substitutions are well-defined.

We first show that requirement (1) holds. For each $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}[\![\theta]\!][\![\theta_r]\!]] \in \Xi[\![\theta]\!][\![\theta_r]\!]$ there must be $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi$ which by the well-formedness of $\mathcal{S}$ is such that $\mathbb{N}_\Gamma \subseteq \mathbb{N}$ and $\mathbb{N} \setminus \mathbb{N}_\Gamma; \Psi \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty has a derivation. From the former it is obvious that the requirement $\mathbb{N}_\Gamma \subseteq (\mathbb{N} \cup supp(\theta))$ is satisfied. The substitution compatibility of $\langle \theta, \Psi' \rangle$ with $\mathcal{S}$ entails that θ is type preserving with respect to $supp(\theta) \cup \Theta_0 \cup \Psi'$, and that $ctx(\theta)$ and $\Psi' \setminus ctx(\theta)$ agree with Ψ on the type assignments to the variables that are common to them. Through an application of Theorem 4.2 there must be a derivation then for $\mathbb{N} \setminus \mathbb{N}_\Gamma; ctx(\theta) \uplus ((\Psi \setminus ctx(\theta)) \cup \Psi') \vdash \mathcal{C}[\mathcal{G}]$ ctx-ty. Using Theorem 4.3 we determine that $(\mathbb{N} \setminus \mathbb{N}_\Gamma) \cup supp(\theta); (\Psi \setminus ctx(\theta)) \cup \Psi') \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!]]$ ctx-ty has a derivation. The substitution compatibility of $\langle \theta, \Psi' \rangle$ with $\mathcal{S}$ further entails that $supp(\theta)$ is disjoint from $\mathbb{N}_\Gamma$, and therefore $(\mathbb{N} \cup supp(\theta)) \setminus \mathbb{N}_\Gamma = (\mathbb{N} \setminus \mathbb{N}_\Gamma) \cup supp(\theta)$. Since θ_r is clearly type preserving with respect to $supp(\theta) \cup \Psi''$, by definition, $ctx(\theta_r)$ and Ψ'' are disjoint, and $ctx(\theta_r) = (\Psi \setminus ctx(\theta)) \cup \Psi''$, there is a derivation for $(\mathbb{N} \cup supp(\theta)) \setminus \mathbb{N}_\Gamma; ctx(\theta_r) \uplus \Psi'' \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!]]$ ctx-ty by Theorem 4.2. Noting that $supp(\theta_r) \subseteq (\mathbb{N} \cup supp(\theta)) \setminus \mathbb{N}_\Gamma$, an application of Theorem 4.3 will determine that $(\mathbb{N} \cup supp(\theta)) \setminus \mathbb{N}_\Gamma; \Psi'' \vdash \mathcal{C}[\mathcal{G}[\![\theta]\!][\![\theta_r]\!]]$ ctx-ty must have a derivation.

We now show that requirement (2) holds. It should be obvious that Ξ^- is the same as $\Xi[\![\theta]\!][\![\theta_r]\!]^-$ as these substitutions do not change the context variables of the sequent. For any formula $F'[\![\theta]\!][\![\theta_r]\!] \in \{F[\![\theta]\!][\![\theta_r]\!]\} \cup \Omega[\![\theta]\!][\![\theta_r]\!]$, the well-formedness of $\mathcal{S}$ ensure there must exist a derivation for $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla*. It follows from the substitution compatibility of $\langle\theta, \Psi'\rangle$ for $\mathcal{S}$ that θ is arity type preserving with respect to the arity typing context $\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup (\Psi \setminus ctx(\theta)) \cup \Psi'$. Since $ctx(\theta)$ and Ψ agree on the type assignments to common variables, we can conclude, using Theorem 3.2, that there must be a derivation for the judgement $ctx(\theta) \uplus (\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup (\Psi \setminus ctx(\theta)) \cup \Psi')\,; \Xi^- \vdash F'$ *fmla*. It follows then, from Theorem 3.4, that $F'[\![\theta]\!]$ is well-defined and there is a derivation for $\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup (\Psi \setminus ctx(\theta)) \cup \Psi'; \Xi \vdash F'[\![\theta]\!]$ *fmla*. The raising substitution θ_r is type preserving with respect to $supp(\theta) \cup \Psi''$ by its construction and so is also type preserving with respect to $\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup \Psi''$. Again using Theorem 3.2 we adjust the arity typing context from the formation judgement for $F'[\![\theta]\!]$ to the form $(\Psi \setminus ctx(\theta) \cup \Psi') \uplus (\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup \Psi'')$ and noting that $ctx(\theta_r)$ will be equal to $\Psi \setminus ctx(\theta) \cup \Psi'$, a second application of Theorem 3.4 will let us conclude $F'[\![\theta]\!][\![\theta_r]\!]$ is well-defined and $\mathbb{N} \cup supp(\theta) \cup \Theta_0 \cup \Psi''; \Xi^- \vdash F'[\![\theta]\!][\![\theta_r]\!]$ *fmla* must have a derivation. □

We will also need to consider the application of substitutions for context variables to sequents. To be meaningfully applied, the context expressions being substituted for the variables must be well-formed with respect to the types associated with the variables. In contrast to term variable substitutions, context variable substitutions are not permitted to introduce new term variables into the sequent and they may use nominal constants that are already present. These notions are formalized below in the notion of appropriate substitutions.

Definition 4.6 (Appropriate Context Variable Substitutions)**.** Let σ be the context variable substitution $\{G_1/\Gamma_1, \ldots, G_n/\Gamma_n\}$. We write Ξ_σ to denote the context Ξ trimmed so as not to include context variables which are in the domain of σ. We say that σ is appropriate for a context variable context Ξ with respect to an arity context Ψ if, for $1 \leq i \leq n$, it is the case that $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ and $supp(\sigma) \setminus \mathbb{N}_i; \Psi; \Xi_\sigma \vdash \mathcal{C}_i[\mathcal{G}_i] \leadsto_{csty} G_i$ has a derivation. The substitutions σ is additionally said to cover Ξ if $\Xi_\sigma = \emptyset$.

Lifting this definition to sequents we say that σ is appropriate for a well-formed sequent $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ if it is appropriate for Ξ with respect to Ψ.

Context types are evidently unaffected by context variable substitutions. Context expressions are impacted by such substitutions but, for the right kind of substitution, they continue to be instances of relevant context types. This is made precise in the theorem below.

Theorem 4.6. *Let σ be a context variable substitution which is appropriate for Ξ with respect to Ψ and such that $supp(\sigma) \subseteq \mathbb{N}$. If $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G$ has a derivation, then $\mathbb{N}; \Psi; \Xi_\sigma \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G[\sigma]$ also has a derivation.*

Proof. This proof is by induction on the derivation of $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} G$. When G is empty the result is obvious, and when it is of the form (G_1, G_2) it follows from application of the inductive hypothesis. When G is a context variable Γ then there must exist some $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}_\Gamma[\mathcal{G}_\Gamma] \in \Xi$. Further, it must be that either $\Gamma[\sigma] = \Gamma$ and $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}_\Gamma[\mathcal{G}_\Gamma] \in \Xi_\sigma$, or $\Gamma[\sigma] = G_i$ and, by the appropriateness of σ, $supp(\sigma) \setminus \mathbb{N}_\Gamma; \Psi; \Xi_\sigma \vdash \mathcal{C}_\Gamma[\mathcal{G}_\Gamma] \leadsto_{csty} G_i$ has a derivation. Since $supp(\sigma) \subseteq \mathbb{N}$, we can infer that $supp(\sigma) \setminus \mathbb{N}_\Gamma \subseteq \mathbb{N} \setminus \mathbb{N}_\Gamma$ and so by Theorem 4.2 there must be a derivation of $\mathbb{N}; \Psi; \Xi_\sigma \vdash \mathcal{C}[\mathcal{G}] \leadsto_{csty} \Gamma[\sigma]$ for any such Γ. $\square$

As with the term substitutions the application of context substitutions may introduce new nominal constants, and we use the technique of raising to permit these constants in substitutions for eigenvariables in the resulting sequent. We formalize the application of a context variable substitution to a well-formed sequent when the conditions of appropriateness are met in the following definition. We assume here and elsewhere that the application of a substitution to a set of formulas distributes to each member of the set.

Definition 4.7 (Applying a Context Substitution to a Sequent)**.** Let $\mathcal{S}$ be a well-formed sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ and σ be appropriate for $\mathcal{S}$. Further let Ψ' be a version of Ψ raised over $supp(\sigma)$ and let θ_r be the corresponding raising substitution. Then the application of σ to $\mathcal{S}$, denoted by $\mathcal{S}[\sigma]$, is the sequent $\mathbb{N} \cup supp(\sigma); \Psi'; \Xi_\sigma[\![\theta_r]\!]; \Omega[\sigma][\![\theta_r]\!] \longrightarrow F[\sigma][\![\theta_r]\!]$.

The following theorem is the counterpart of Theorem 4.5 for context variable substitutions.

Theorem 4.7. *Let $\mathcal{S}$ be a well-formed sequent and let σ be a context variable substitution that is appropriate for $\mathcal{S}$. Then $\mathcal{S}[\sigma]$ is well-defined and is a well-formed sequent.*

Proof. Suppose that $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ is a well-formed sequent and σ an appropriate context variable substitution for $\mathcal{S}$. Following Definition 4.7, let Ξ' be the context variable context Ξ without the context variables substituted for by σ, Ψ' be a version of Ψ raised over $supp(\sigma) \setminus \mathbb{N}$, and θ_r the corresponding raising substitution. We must then show the following:

1. for each $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}[\![\theta_r]\!]] \in \Xi'[\![\theta_r]\!]$, it is the case that $\mathbb{N}_\Gamma \subseteq (\mathbb{N} \cup supp(\sigma))$ and $(\mathbb{N} \cup supp(\sigma)) \setminus \mathbb{N}_\Gamma; \Psi' \vdash \mathcal{C}[\mathcal{G}[\![\theta_r]\!]]$ ctx-ty has a derivation, and

2. for each formula $F'[\sigma][\![\theta_r]\!] \in \{F[\sigma][\![\theta_r]\!]\} \cup \Omega[\sigma][\![\theta_r]\!]$, it is the case that the judgement $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi'; \Xi'[\![\theta_r]\!]^- \vdash F'[\sigma][\![\theta_r]\!]$ *fmla* has a derivation.

In showing these two requirements we will also ensure that the relevant substitutions are well-defined.

Using an argument similar to that in the proof of Theorem 4.5, we can show that $(\mathbb{N} \cup supp(\sigma)) \setminus \mathbb{N}_\Gamma; \Psi' \vdash \mathcal{C}[\mathcal{G}[\![\theta_r]\!]]$ ctx-ty has a derivation for each $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}[\![\theta_r]\!]] \in \Xi'[\![\theta_r]\!]$.

We now show that requirement (2) holds. Note that $\Xi'[\![\theta_r]\!]^-$ is the same collection as Ξ'^-. For any formula $F'[\sigma][\![\theta_r]\!] \in \{F[\sigma][\![\theta_r]\!]\} \cup \Omega[\sigma][\![\theta_r]\!]$ it must be that $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* has a derivation by the well-formedness of $\mathcal{S}$. Thus by Theorem 3.2 we can conclude that $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* must be derivable. The appropriateness of σ for $\mathcal{S}$ and an application of Theorem 4.1 ensure that for all $G/\Gamma \in \sigma$, there is a derivation of $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi; \Xi'^- \vdash G$ *context*. We can then conclude from an application of Theorem 3.4 that there must be a derivation of $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi; \Xi'^- \vdash F'[\sigma]$ *fmla*.

Again extending the arity typing context, using Theorem 3.2 there must be a derivation for $\Psi \uplus (\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi'); \Xi'^{-} \vdash F'[\sigma]\ fmla$. Recalling that θ_r is arity type preserving with respect to $(supp(\sigma) \setminus \mathbb{N}) \cup \Psi'$, and thus with respect to $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi'$, and that $ctx(\theta_r) = \Psi$, Theorem 3.4 allows us conclude that $F'[\sigma][\![\theta_r]\!]$ is well-defined and that $\mathbb{N} \cup supp(\sigma) \cup \Theta_0 \cup \Psi'; \Xi'^{-} \vdash F'[\sigma][\![\theta_r]\!]\ fmla$ has a derivation $\square$

We are now in a position to define validity for sequents. For a sequent containing term and context variables, this is done by considering all their relevant substitution instances. For a sequent devoid of variables, we base the definition on the validity of closed formulas.

Definition 4.8. A well-formed sequent of the form $\mathbb{N}; \emptyset; \emptyset; \Omega \longrightarrow F$ is valid if whenever all the formulas in Ω are valid, F is a valid formula; well-formed sequents of this form are referred to as *closed* sequents. A well-formed sequent $\mathcal{S}$ of the form $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ is valid if for every term substitution θ that is type preserving with respect to $\mathcal{N} \cup \Theta_0$ and such that $\Psi = ctx(\theta)$ and $\langle \theta, \emptyset \rangle$ is substitution compatible with $\mathcal{S}$, and for every context substitution σ that is appropriate and closed for $\mathcal{S}[\![\theta]\!]_\emptyset$, it is the case that $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ is valid. Note that each such $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ will be a well-formed and closed sequent in these circumstances and we shall refer to it as the closed instance of $\mathcal{S}$ identified by θ and σ.

The following theorem, whose proof is obvious, provides the basis for using our proof system for determining the validity of formulas.

Theorem 4.8. *Let F be a formula such that $\mathcal{N} \cup \Theta_0; \emptyset \vdash F\ fmla$ is derivable and let $\mathbb{N}$ be the set of nominal constants that appear in F. Then the sequent $\mathbb{N}; \emptyset; \emptyset; \emptyset \longrightarrow F$ is well-formed. Moreover, F is valid if and only if $\mathbb{N}; \emptyset; \emptyset; \emptyset \longrightarrow F$ is.*

In Section 3.4, we had noted an invariance of validity for formulas under permutations of nominal constants. We observe an analogous property concerning sequents. We first explain what it means to apply a permutation to a sequent.

Definition 4.9 (Applying Permutations of Nominal Constants to Sequents)**.** The application of a permutation π to a context variable type distributes to the constituent block instances and the application to a collection of formulas or a context variable context distributes to the members of the collection. The application to a sequent $\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F$ yields the sequent $\mathbb{N}';\Psi;\pi.\Xi';\pi.\Omega' \longrightarrow \pi.F'$ where $\mathbb{N}' = \{n' |\ n \in \mathbb{N} \text{ and } \pi(n) = n'\}$.

Some useful results about the interaction of permutations with sequents are given below. Theorems 4.9 and 4.10 consider the composition of permutations and substitutions. Their proofs are straightforward, relying on the invariance of the derivability of judgements under permutation which can be verified easily by induction on such derivations. Theorem 4.11 is the analogous property of Theorem 3.6, and it's proof is straightforward given that result and the definition of validity.

Theorem 4.9. *Let π be a permutation of the nominal constants, let $\mathcal{S}$ be a well-formed sequent and let $\langle\theta,\Psi\rangle$ be substitution compatible with $\pi.\mathcal{S}$. Then $\langle\pi^{-1}.\theta,\Psi\rangle$ is substitution compatible with $\mathcal{S}$ and $(\pi.\mathcal{S})[\![\theta]\!]_\Psi = \pi.(\mathcal{S}[\![\pi^{-1}.\theta]\!]_\Psi)$.*

Theorem 4.10. *Let π be a permutation of the nominal constants, let $\mathcal{S}$ be a well-formed sequent and let σ be a context variable substitution that is appropriate for $\pi.\mathcal{S}$. Then $\pi^{-1}\sigma$ is appropriate for $\mathcal{S}$ and $(\pi.\mathcal{S})[\sigma] = \pi.(\mathcal{S}[\pi^{-1}.\sigma])$.*

Theorem 4.11. *If π is a permutation of the nominal constants and $\mathcal{S}$ is a well-formed closed sequent that is valid, then $\pi.\mathcal{S}$ is also a closed, valid sequent.*

4.2 The Core Proof Rules

We consider in this section a collection of proof rules that internalize the interpretation of the logical symbols that may appear in formulas and also some properties that flow from the meanings of sequents. More specifically, the first subsection below presents some structural rules, the second subsections identifies axioms and the cut rule, and the third subsection introduces rules for the logical symbols. Beyond presenting the rules, we are interested in showing that the rules are sound and that in their context we may limit our attention to only well-formed sequents in trying to construct a derivation for a well-formed sequent. The

$$\frac{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1}{\mathbb{N};\Psi;\Xi;\Omega,F_2 \longrightarrow F_1}\ \mathit{weak} \qquad \frac{\mathbb{N};\Psi;\Xi;\Omega,F_2,F_2 \longrightarrow F_1}{\mathbb{N};\Psi;\Xi;\Omega,F_2 \longrightarrow F_1}\ \mathit{cont}$$

$$\frac{\begin{array}{c}\Xi = \{\Gamma_i{\uparrow}(\mathbb{N}_i \setminus \mathbb{N}') : \mathcal{C}_i[\mathcal{G}_i] \mid \Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \overline{\Xi}\}\\ \{(\mathbb{N},\mathbb{N}') \setminus \mathbb{N}_i; (\Psi,\Psi') \vdash \mathcal{C}_i[\mathcal{G}_i] \text{ ctx-ty} \mid \Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi'\}\\ \mathbb{N},\mathbb{N}';\Psi,\Psi';\overline{\Xi},\Xi';\Omega \longrightarrow F\end{array}}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F}\ \mathit{ctx\text{-}str}$$

$$\frac{\begin{array}{c}\Xi = \{\Gamma_i{\uparrow}(\mathbb{N}_i \setminus \mathbb{N}') : \mathcal{C}_i[\mathcal{G}_i] \mid \Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \overline{\Xi}\}\\ \{(\mathbb{N} \setminus \mathbb{N}_i); \Psi \vdash \mathcal{C}_i[\mathcal{G}_i] \text{ ctx-ty} \mid \Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi\}\\ \{\mathbb{N} \cup \Theta_0 \cup \Psi; \{\Gamma \mid \Gamma{\uparrow}\mathbb{N} : \mathcal{C}[\mathcal{G}] \in \Xi\} \vdash F'\ \mathit{fmla} \mid F' \in \Omega \cup \{F\}\}\\ \mathbb{N};\Psi;\Xi;\Omega \longrightarrow F\end{array}}{\mathbb{N},\mathbb{N}';\Psi,\Psi';\overline{\Xi},\Xi';\Omega \longrightarrow F}\ \mathit{ctx\text{-}wk}$$

Figure 4.2: The Structural Rules

latter property is verified by showing that the premise sequents of each rule are well-formed if the conclusion sequent is.

4.2.1 Structural Rules

This subcollection of rules is presented in Figure 4.2. These rules can be subcategorized into those that allow for weakening and contracting the assumption set in a sequent and those that permit the weakening and strengthening of the support set, the eigenvariable context, and the context variable context. Rules of the second subcategory encode the fact that vacuous (well-formed) extensions to the bindings manifest in a sequent will not impact its validity. The strengthening and weakening rules for contexts include premises that force modifications to context variable types and the satisfaction of typing judgements that are necessary to ensure the well-formedness of the sequents in any application of the rule.

The following theorem shows that these rules require the proof of only well-formed sequents in constructing a proof of a well-formed sequent.

Theorem 4.12. *The following property holds for each rule in Figure 4.2: if the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then all the sequent premises must be well-formed.*

Proof. We consider each rule described in Figure 4.2.

Case: *weak*

For a well-formed conclusion sequent $\mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F_1$ we must show that the premise sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_1$ is also well-formed. Since the context variable contexts are the same in both sequents, the goal formulas are the same, and $\Omega \subseteq \Omega \cup \{F_2\}$ we can easy determine the well-formedness of the premise by applying the definition of well-formedness to the conclusion sequent.

Case: *cont*

Suppose that the sequent $\mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F_1$ is well-formed. Noting that the only addition to the premise sequent is a copy of the well-formed formula F_2 to the assumption set, and otherwise the two sequents are identical, it is clear that the premise sequent must also be valid.

Case: *ctx-str*

Suppose that a sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ is well-formed, the context variable context Ξ is of the form described in the rule, and for each $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi'$ the judgement $((\mathbb{N}, \mathbb{N}') \setminus \mathbb{N}_i)\,; (\Psi, \Psi') \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty has a derivation. By Theorem 3.2 it is clear that the well-formedness of the formulas in $\Omega \cup \{F\}$ is preserved by the extensions to the support set, arity typing context, and context variable context. The premises of this rule ensure that the context variable types in Ξ' are well-formed, and so it only remains to conclude those in $\overline{\Xi}$ are as well. For each $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \overline{\Xi}$ there is a $\Gamma_i \uparrow (\mathbb{N}_i \setminus \mathbb{N}') : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ for which there is a derivation of $(\mathbb{N} \setminus (\mathbb{N}_i \setminus \mathbb{N}'))\,; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty by the well-formedness of the conclusion sequent. So by Theorem 4.2, $(\mathbb{N} \setminus \mathbb{N}_i)\,; \Psi, \Psi' \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty is derivable for each entry in $\overline{\Xi}$. Therefore the premise sequent must be well-formed.

Case: *ctx-wk*

The well-formedness of $\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F$ is obvious given the sets of well-formedness derivations in the premises. $\square$

The argument for soundness has an intuitively obvious structure in all the cases other than when the support set is affected. In the case when the support set is expanded, the reasoning is still straightforward and is based on observing that the instances of the weakened form of the sequent will be a subset of the instances of the premise sequent. When the support set is smaller in the conclusion sequent, the argument is a little more subtle: we must use the fact that permutations of nominal constants that do not appear in the context types or the formulas in a sequent do not impact on validity. This observation is embedded in the following lemma.

Lemma 4.1. *Let* $\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F$ *and* $\mathbb{N}';\Psi';\Xi';\Omega \longrightarrow F$ *be well-formed sequents such that* $\mathbb{N}' \subseteq \mathbb{N}$*,* $\Psi' \subseteq \Psi$*, and there is some subset* $\overline{\Xi}$ *for the context variable context* Ξ *where* $\Xi' = \left\{\Gamma_i\uparrow(\mathbb{N}_i \setminus (\mathbb{N}\setminus\mathbb{N}')) : \mathcal{C}_i[\mathcal{G}_i] \;\middle|\; \Gamma_i\uparrow\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \overline{\Xi}\right\}$*. Then the sequent* $\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F$ *will be valid if and only if* $\mathbb{N}';\Psi';\Xi';\Omega \longrightarrow F$ *is valid.*

Proof. Let $\mathcal{S}$ denote the sequent $\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F$ and $\mathcal{S}'$ the sequent $\mathbb{N}';\Psi';\Xi';\Omega \longrightarrow F$. Given that both $\mathcal{S}$ and $\mathcal{S}'$ are well-formed sequents, no context variables in $(\Xi\setminus\Xi')$ appear in the formula F or the formulas in Ω, no variables in $(\Psi\setminus\Psi')$ appear in the context variable context Ξ', the formula F, or the formulas in Ω, and no nominal constants in $(\mathbb{N}\setminus\mathbb{N}')$ appear in the context variable context Ξ', the formula F, or the formulas in Ω.

We consider each direction of the implication.

Case: $(\Rightarrow)$

Let θ and σ identify an arbitrary closed instance of the sequent $\mathcal{S}'$. Then $\langle\theta,\emptyset\rangle$ is substitution compatible for $\mathcal{S}'$ and σ is appropriate for $\mathcal{S}'[\![\theta]\!]_{\emptyset}$. So $\mathrm{supp}\,(\theta)\cap\mathbb{N}' = \emptyset$, but it is possible that $\mathrm{supp}\,(\theta)\cap\mathbb{N}$ is not empty. Let π be a permutation which maps nominal constants in $(\mathbb{N}\setminus\mathbb{N}')$ to some new names not appearing in $\mathbb{N}$, θ, or σ. Then $\langle\pi.\theta,(\Psi\setminus\Psi')\rangle$ will be substitution compatible for the sequent $\mathcal{S}$. And since the restricted names for context variables in $\overline{\Xi}$

only differ from the restricted sets of names from Ξ' by nominal constants appearing in $\mathbb{N} \setminus \mathbb{N}'$ the substitution $\pi.\sigma$ will be appropriate for $\mathcal{S}[\![\pi.\theta]\!]_{(\Psi\setminus\Psi')}$.

So consider the sequent $\mathcal{S}[\![\pi.\theta]\!]_{(\Psi\setminus\Psi')}[\pi.\sigma]$. Let θ' and σ' identify an arbitrary closed instance of this sequent. This closed sequent must be the same as the closed instance of $\mathcal{S}$ identified by $(\theta' \circ \pi.\theta)$ and $(\sigma' \circ \pi.\sigma)$ given that θ', θ, σ', and σ are all closed substitutions. Since $\mathcal{S}$ is valid by assumption, this particular closed instance identified by $(\theta' \circ \pi.\theta)$ and $(\sigma' \circ \pi.\sigma)$ will thus be valid. But we know that no context variables in $\Xi \setminus \Xi'$ or variables in $\Psi \setminus \Psi'$ appear in F or any formula in Ω. Further, both θ and σ are closed substitutions and so these variables also cannot appear in $F[\![\pi.\theta]\!][\pi.\sigma]$ or any formula in $\Omega[\![\pi.\theta]\!][\pi.\sigma]$. Thus the closed sequent $(\mathcal{S}[\![\pi.\theta]\!]_{(\Psi\setminus\Psi')}[\pi.\sigma])[\![\theta']\!]_{\emptyset}[\sigma']$ is in fact equivalent to the closed sequent $\mathcal{S}'[\![\pi.\theta]\!]_{(\Psi\setminus\Psi')}[\pi.\sigma]$ which must therefore be a valid closed sequent. Since π was constructed such that it does not permute any of the nominal constants in $\mathbb{N}'$ this is the same as the closed sequent $\pi.(\mathcal{S}'[\![\theta]\!][\sigma])$. By Theorem 4.11, validity of closed sequents is preserved by permutations and thus we can conclude that $\mathcal{S}'[\![\theta]\!][\sigma]$ is valid.

Since all closed instances of $\mathcal{S}'$ must therefore be valid, this sequent is valid.

Case: ($\Leftarrow$)

Let θ and σ identify an arbitrary closed instance of the sequent $\mathcal{S}$. Then $\langle\theta, \emptyset\rangle$ is substitution compatible with $\mathcal{S}$ and σ is appropriate for $\mathcal{S}[\![\theta]\!]_{\emptyset}$. If some formula in $\Omega[\![\theta]\!][\sigma]$ is not valid then $\mathcal{S}[\![\theta]\!]_{\emptyset}[\sigma]$ is vacuously valid. So suppose instead that all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid.

Given that $\mathbb{N}' \subseteq \mathbb{N}$ and $\Psi' \subseteq \Psi$ this $\langle\theta, \emptyset\rangle$ is also substitution compatible with $\mathcal{S}'$. Further, because $dom\,(\Xi') \subseteq dom\,(\Xi)$ and the annotation sets on variables in Ξ' will all be subsets of the annotation set for that context variable in Ξ the substitution σ will be appropriate for $\mathcal{S}'[\![\theta]\!]_{\emptyset}$. Given the validity of $\mathcal{S}'$, the closed instance $\mathcal{S}'[\![\theta]\!]_{\emptyset}[\sigma]$ must be valid. Given the assumption that all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid, the validity of this closed sequent means $F[\![\theta]\!][\sigma]$ must be valid. Thus $\mathcal{S}[\![\theta]\!]_{\emptyset}[\sigma]$ is valid in this case as well.

Since all closed instances of the sequent $\mathcal{S}$ are valid, it is a valid sequent. $\square$

We may now establish the soundness of the structural rules.

Theorem 4.13. *The following property holds for every instance of each of the rules in Figure 4.2: if the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and the premise sequent is valid, then the conclusion sequent must also be valid.*

Proof. We consider each rule described in Figure 4.2.

Case: *weak*

Given the structure of the sequents, it is clear that any substitutions identifying a closed instance of the conclusion sequent will also identify a closed instance of the premise sequent. Since $\Omega \subseteq \Omega \cup \{F_2\}$, it is also clear that for any substitutions θ and σ identifying a closed instance of the conclusion sequent, the formula $F_1[\![\theta]\!][\sigma]$ must be valid whenever every formula in $\Omega[\![\theta]\!][\sigma] \cup \{F_2[\![\theta]\!][\sigma]\}$ is valid. But then every closed instance of the sequent $\mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F_1$ is valid, and thus the sequent itself is valid.

Case: *cont*

It is clear from the structure of the sequents that any θ and σ identifying a closed instance of the conclusion sequent will also identify a closed instance of the premise sequent. Furthermore, the collection of formulas $\Omega[\![\theta]\!][\sigma], F_2[\![\theta]\!][\sigma], F_2[\![\theta]\!][\sigma]$ are clearly all valid whenever the collection of formulas $\Omega[\![\theta]\!][\sigma], F_2[\![\theta]\!][\sigma]$ are all valid. Therefore from the validity of the premise sequent we can conclude that the conclusion sequent is also valid as every closed instance of this sequent must be valid.

Case: *ctx-str* **or** *ctx-wk*

Both cases are resolved through an application of Lemma 4.1 given the well-formedness of the conclusion and premise sequents. □

4.2.2 The Axiom and the Cut Rule

The two rules of interest here are also related to the interpretation of sequents but focus more specifically on the logical relationship between the formula collections. The *cut* rule codifies the notion of lemmas: if we can show the validity of a formula relative to a given assumption set, then this formula can be included in the assumptions to simplify the reasoning process.

The *id* rule recognizes the validity of a sequent in which the conclusion formula appears in the assumption set.

In its simplest form, the *id* rule would require the conclusion formula to be included as is in the assumption set, possibly with a renaming of the bound variables that appear in it. It is possible, and also pragmatically useful, to generalize this form to allow also for a permutation of nominal constants in the formulas in the process of matching. However, this has to be done with care to ensure that identity under the considered permutations continues to hold even after later instantiations of term and context variables appearing in the formulas. The specific form of equivalence for formulas under permutations that we will use is the content of the following definition.

Definition 4.10 (Formula Equivalence)**.** The equivalence of two context expressions G_1 and G_2 with respect to a context variable context Ξ and a permutation π, written $\Xi \vdash G_2 \equiv_\pi G_1$, is a relation defined by the following three clauses:

1. $\Xi \vdash \cdot \equiv_\pi \cdot$ holds for any Ξ and π.
2. $\Xi \vdash \Gamma \equiv_\pi \Gamma$ holds if $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}[\mathcal{G}] \in \Xi$ and $\mathrm{supp}\,(\pi) \subseteq \mathbb{N}_i$.
3. If $G_1 = (G_1', n_1 : A_1)$ and $G_2 = (G_2', n_2 : A_2)$ then $\Xi \vdash G_2 \equiv_\pi G_1$ holds if $\pi.n_2$ is identical to n_1, $\pi.A_2$ is identical to A_1 (up to a renaming of bound variables), and $\Xi \vdash G_2 \equiv_\pi G_1$ holds.

Two atomic formulas $\{G' \vdash M' : A'\}$ and $\{G \vdash M : A\}$ are considered equivalent with respect to Ξ and π if G' and G are equivalent with respect to this Ξ and π and $\pi.M'$ and M and $\pi.A'$ and A are respectively identical up to a renaming of bound variables. Two arbitrary formulas are considered equivalent with respect to Ξ and π if their component parts are so equivalent, allowing, of course, for a renaming of variables bound by quantifiers. Equivalence of formulas is represented by the judgement $\Xi \vdash F' \equiv_\pi F$.

The *id* and the *cut* rules are presented in Figure 4.3. The *id* rule limits the permutations that can be considered to be ones that rename only nominal constants appearing in the

$$\frac{F' \in \Omega \quad \mathrm{supp}\,(\pi) \subseteq \mathbb{N} \quad \Xi \vdash F' \equiv_\pi F}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F}\ id$$

$$\frac{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_2 \quad \mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F_1 \quad \mathbb{N} \cup \Theta_0 \cup \Psi; dom\,(\Xi) \vdash F_2\ fmla}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_1}\ cut$$

Figure 4.3: The Axiom and the Cut Rule

support set of the sequent. The *cut* rule includes a premise that ensures the wellformedness of the cut formula. The following theorem shows that these require the proofs of only well-formed sequents in constructing a proof of a well-formed sequent.

Theorem 4.14. *The following property holds of the* id *and the* cut *rule: if the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then the premise sequents must be well-formed.*

Proof. The requirement is vacuously true for the *id* rule and it has an obvious proof for the *cut* rule.

□

In showing the soundness of the *id* rule, we will need the observation that the equivalence of formulas modulo permutations is preserved under the kinds of substitutions that have to be considered in determining the validity of sequents. This observation is the content of the two lemmas below.

Lemma 4.2. *Suppose the for some formulas F_1 and F_2, $\Xi \vdash F_2 \equiv_\pi F_1$ is holds. If θ is a hereditary substitution such that $supp\,(\theta) \cap supp\,(\pi) = \emptyset$ and both $F_2[\![\theta]\!] = F_2'$ and $F_1[\![\theta]\!] = F_1'$ have derivations for some F_1' and F_2', then $\Xi[\![\theta]\!] \vdash F_2[\![\theta]\!] \equiv_\pi F_1[\![\theta]\!]$ holds.*

Proof. This is proved by induction on the structure of $\Xi \vdash F_2 \equiv_\pi F_1$. Consider the possible cases for the structure of F_1 (or equivalently, F_2).

The case where the formula is $\top$ or $\bot$, this observation is obvious. The cases when F_1 is $F \wedge F'$, $F \vee F'$, $F \supset F'$, $\forall x : \alpha.F$, $\exists x : \alpha.F$, or $\Pi\,\Gamma : \mathcal{C}.F$ are easily argued with recourse to the induction hypothesis and by noting that the definition of substitution distributes to the component parts.

When F_1 and F_2 are atomic formulas $\{G_1 \vdash M_1 : A_1\}$ and $\{G_2 \vdash M_2 : A_2\}$ respectively, $\pi.M_2 =_\alpha M_1$, $\pi.A_2 =_\alpha A_1$, and $\Xi \vdash G_2 \equiv_\pi G_1$ will be derivable by definition. We observe that for any LF term M (resp. type A), if $M[\![\theta]\!]$ (resp. $A[\![\theta]\!]$) is defined then $M[\![\pi.\theta]\!]$ (resp. $A[\![\pi.\theta]\!]$) is defined and $\pi.(M[\![\theta]\!]) = (\pi.M)[\![\pi.\theta]\!]$ (resp. $\pi.(A[\![\theta]\!]) = (\pi.A)[\![\pi.\theta]\!]$). This observation can be proved easily by induction on the structure of M, and using this result the observation for A proved by induction on A. The support sets of θ and π must be disjoint, thus by this observation $(\pi.A_2)[\![\theta]\!] =_\alpha \pi.(A_2[\![\theta]\!])$ and $(\pi.M_2)[\![\theta]\!] =_\alpha \pi.(M_2[\![\theta]\!])$, and therefore that $\pi.(A_2[\![\theta]\!]) =_\alpha A_1[\![\theta]\!]$ and $\pi.(M_2[\![\theta]\!]) =_\alpha M_1[\![\theta]\!]$. What remains is to show that $\Xi[\![\theta]\!] \vdash G_2[\![\theta]\!] \equiv_\pi G_1[\![\theta]\!]$, which we argue by an induction on the context expression G_1.

If G_1 is $\cdot$ or some Γ_i, then clearly $G_2 = G_1$, and further, $G_1[\![\theta]\!] = G_1$. Thus the equivalence $\Xi[\![\theta]\!] \vdash G_2[\![\theta]\!] \equiv_\pi G_1[\![\theta]\!]$ has an obvious derivation. If the context expression G_1 is of the form $(G_1', n_1' : A_1')$, then G_2 is of the form $(G_2', n_2' : A_2')$ and $\pi.n_2'$ is the same nominal constant as n_1', $\pi.A_2'$ is equal to A_1' up to renaming of bound variables, and $\Xi \vdash G_2' \equiv_\pi G_1'$ is derivable. By induction then, $\Xi[\![\theta]\!] \vdash G_2'[\![\theta]\!] \equiv_\pi G_1'[\![\theta]\!]$ will be derivable. From the assumption that $\mathrm{supp}\,(\theta) \cap \mathrm{supp}\,(\pi) = \emptyset$ we can infer that $\pi.\theta = \theta$. Thus $\pi.(A_2'[\![\theta]\!]) =_\alpha (\pi.A_2')[\![\theta]\!]$ by our earlier observation about permutations on LF types, and so $\pi.(A_2'[\![\theta]\!]) =_\alpha A_1'[\![\theta]\!]$. From this we can construct a derivation for $\Xi[\![\theta]\!] \vdash G_2[\![\theta]\!] \equiv_\pi G_1[\![\theta]\!]$.

Thus we have shown that $\Xi[\![\theta]\!] \vdash G_2[\![\theta]\!] \equiv_\pi G_1[\![\theta]\!]$ holds and therefore can conclude that $\Xi[\![\theta]\!] \vdash F_2[\![\theta]\!] \equiv_\pi F_1[\![\theta]\!]$ must have a derivation. □

Lemma 4.3. *Suppose σ is an appropriate substitution for Ξ with respect to Ψ and that $\Xi \vdash F' \equiv_\pi F$ has a derivation. Then $\Xi_\sigma \vdash F'[\sigma] \equiv_\pi F[\sigma]$ will have a derivation.*

Proof. This is proved by induction on the structure of $\Xi \vdash F_2 \equiv_\pi F_1$. Consider the possible cases for the structure of F_1 (or equivalently, F_2).

The case where the formula is $\top$ or $\bot$, this observation is obvious. The cases when F_1 is $F \wedge F'$, $F \vee F'$, $F \supset F'$, $\forall x : \alpha.F$, $\exists x : \alpha.F$, or $\Pi\,\Gamma : \mathcal{C}.F$ are easily argued with recourse to the induction hypothesis and by noting that the definition of substitution distributes to the component parts.

In the case that F_1 is atomic, then F_1 and F_2 are of the form $\{G_1 \vdash M_1 : A_1\}$ and $\{G_2 \vdash M_2 : A_2\}$ respectively, and $\pi.M_2$ is the same as M_1 up to renaming, $\pi.A_2$ and A_1 are equal up to renaming, and $\Xi \vdash G_2 \equiv_\pi G_1$ is derivable. Given that $\{G \vdash M : A\}\,[\sigma] = \{G[\sigma] \vdash M : A\}$, it only remains to show that $\Xi_\sigma \vdash G_2[\sigma] \equiv_\pi G_1[\sigma]$ has a derivation to conclude $\Xi_\sigma \vdash F_2[\sigma] \equiv_\pi F_1[\sigma]$ is derivable. We prove that $\Xi_\sigma \vdash G_2[\sigma] \equiv_\pi G_1[\sigma]$ is derivable by induction on the structure of $\Xi \vdash G_2 \equiv_\pi G_1$.

When $G_1 = G_2 = \cdot$ this result is obvious. When $G_1 = (G_1', n_1' : A_1')$ this is easily argued with recourse to the inductive hypothesis. When G_1 is some context variable Γ_i, then $G_2 = \Gamma_i$, $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$, and $\mathrm{supp}\,(\pi) \subseteq \mathbb{N}_i$. If Γ_i is not in the domain of σ, then clearly $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi_\sigma$ and $\Xi_\sigma \vdash G_2[\sigma] \equiv_\pi G_1[\sigma]$ has an obvious derivation. If instead Γ_i is in the domain of σ then there is some G_i such that $G_2[\sigma] = G_1[\sigma] = G_i$ and by the appropriateness of σ, $\mathit{supp}(\sigma) \setminus \mathbb{N}_i; \Psi; \Xi_\sigma \vdash \mathcal{C}_i[\mathcal{G}_i] \rightsquigarrow_{\mathit{csty}} G_i$ must have a derivation. By an obvious inductive argument we observe that because $\mathrm{supp}\,(\pi) \subseteq \mathbb{N}_i$, $\pi.G_i = G_i$ and there is a derivation of $\Xi_\sigma \vdash G_2[\sigma] \equiv_\pi G_1[\sigma]$, as needed.

From this we can conclude that there will be a derivation for $\Xi_\sigma \vdash F'[\sigma] \equiv_\pi F[\sigma]$. $\Box$

We can now show the soundness of the *id* and *cut* rules.

Theorem 4.15. *The following property holds for every instance of the id and cut rules: if the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and all the premise sequents are valid, then the conclusion sequent must also be valid.*

Proof. Consider the case for each rule.

Case: *id*

By assumption, for goal formula F, assumption formula $F' \in \Omega$, and permutation π, π is a permutation of some subset of nominal constants in $\mathbb{N}$ and $\Xi \vdash F' \equiv_\pi F$ has a derivation. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid, this closed instance would be vacuously valid, so assume that all such formulae are valid. In particular then, $F'[\![\theta]\!][\sigma]$ will be valid. The substitution compatibility of $\langle\theta, \emptyset\rangle$ and the appropriateness of σ are sufficient to satisfy the requirements of Lemmas 4.3 and 4.2. Thus there must be a derivation of $\emptyset \vdash F'[\![\theta]\!][\sigma] \equiv_\pi F[\![\theta]\!][\sigma]$. But then by Theorem 3.6, $F[\![\theta]\!][\sigma]$ is also valid, and therefore the conclusion sequent will be valid.

Case: *cut*

By assumption, the formula F' is such that the premise sequents $\mathcal{S}_1 = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F'$ and $\mathcal{S}_2 = \mathbb{N}; \Psi; \Xi; \Omega, F' \longrightarrow F$ are both valid. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ; these substitutions clearly also identify closed instances of the premise sequents $\mathcal{S}_1$ and $\mathcal{S}_2$. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid, then this closed instance would be vacuously valid. If all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid, then by the validity of $\mathcal{S}_1$ the formula $F'[\![\theta]\!][\sigma]$ must be valid. But then all formulas in $(\Omega, F')[\![\theta]\!][\sigma]$ are valid, and by the validity of $\mathcal{S}_2$ the goal formula $F[\![\theta]\!][\sigma]$ must be valid. Therefore the conclusion sequent of the *cut* rule must be valid. □

4.2.3 Rules for the Logical Symbols

Figure 4.4 presents derivation rules that are based on the meanings of the logical symbols that can appear in formulas. In the application of these rules, we assume the equivalence of formulas under the renaming of quantified variables.

As in the previous cases, we show that these rules also allow for a limitation of attention to well-formed sequents.

Theorem 4.16. *The following property holds of the rules in Figure 4.4: if the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then the premise sequents must be well-formed.*

$$\frac{}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \top}\ \top\text{-}R \qquad \frac{}{\mathbb{N};\Psi;\Xi;\Omega,\bot \longrightarrow F}\ \bot\text{-}L$$

$$\frac{\mathbb{N};\Psi;\Xi,\Gamma'\!\uparrow\!\emptyset : \mathcal{C}[\cdot];\Omega \longrightarrow F[\Gamma'/\Gamma] \quad \Gamma' \notin dom\,(\Xi)}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \Pi\,\Gamma : \mathcal{C}.F}\ \Pi\text{-}R$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega,F_1[G/\Gamma] \longrightarrow F_2 \quad \mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\cdot] \rightsquigarrow_{csty} G}{\mathbb{N};\Psi;\Xi;\Omega,\Pi\,\Gamma : \mathcal{C}.F_1 \longrightarrow F_2}\ \Pi\text{-}L$$

$$\frac{\begin{array}{cc}\mathbb{N} = \{n_1:\alpha_1,\ldots,n_m:\alpha_m\} & y \notin dom\,(\Psi)\\ \mathbb{N};\Psi \cup \{y : (\alpha_1 \to \ldots \to \alpha_m \to \alpha)\};\Xi;\Omega \longrightarrow F' & F[\![\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}]\!] = F'\end{array}}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \forall x : \alpha.F}\ \forall\text{-}R$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega,F_1' \longrightarrow F_2 \quad F_1[\![\{\langle x,t,\alpha\rangle\}]\!] = F_1' \quad \mathbb{N}\cup\Theta_0\cup\Psi \vdash_{at} t : \alpha}{\mathbb{N};\Psi;\Xi;\Omega,\forall x:\alpha.F_1 \longrightarrow F_2}\ \forall\text{-}L$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F' \quad F[\![\{\langle x,t,\alpha\rangle\}]\!] = F' \quad \mathbb{N}\cup\Theta_0\cup\Psi \vdash_{at} t : \alpha}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \exists x:\alpha.F}\ \exists\text{-}R$$

$$\frac{\begin{array}{c}\mathbb{N} = \{n_1:\alpha_1,\ldots,n_m:\alpha_m\} \quad y \notin dom\,(\Psi) \quad F_1[\![\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}]\!] = F_1'\\ \mathbb{N};\Psi \cup \{y : (\alpha_1 \to \ldots \to \alpha_m \to \alpha)\};\Xi;\Omega,F_1' \longrightarrow F_2\end{array}}{\mathbb{N};\Psi;\Xi;\Omega,\exists x:\alpha.F_1 \longrightarrow F_2}\ \exists\text{-}L$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1 \quad \mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_2}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1 \wedge F_2}\ \wedge\text{-}R \quad \frac{\mathbb{N};\Psi;\Xi;\Omega,F_i \longrightarrow F \quad i \in \{1,2\}}{\mathbb{N};\Psi;\Xi;\Omega,F_1 \wedge F_2 \longrightarrow F}\ \wedge\text{-}L_i$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_i \quad i \in \{1,2\}}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1 \vee F_2}\ \vee\text{-}R_i \quad \frac{\mathbb{N};\Psi;\Xi;\Omega,F_1 \longrightarrow F \quad \mathbb{N};\Psi;\Xi;\Omega,F_2 \longrightarrow F}{\mathbb{N};\Psi;\Xi;\Omega,F_1 \vee F_2 \longrightarrow F}\ \vee\text{-}L$$

$$\frac{\mathbb{N};\Psi;\Xi;\Omega,F_1 \longrightarrow F_2}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1 \supset F_2}\ \supset\text{-}R \qquad \frac{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow F_1 \quad \mathbb{N};\Psi;\Xi;\Omega,F_2 \longrightarrow F}{\mathbb{N};\Psi;\Xi;\Omega,F_1 \supset F_2 \longrightarrow F}\ \supset\text{-}L$$

Figure 4.4: The Logical Rules

Proof. Consider as cases each rule in Figure 4.4.

Case: $\top$-R and $\bot$-L.

There are no sequents in the premises of these rules so the property will clearly hold.

Case: Π-R

By the well-formedness of the conclusion sequent

1. for each $\Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$, $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty has a derivation,

2. $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \Pi\,\Gamma : \mathcal{C}.F$ *fmla* has a derivation, and

3. for each $F' \in \Omega$, $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* has a derivation.

From (1) and the obvious derivation for $\mathbb{N}; \Psi \vdash \mathcal{C}[\cdot]$ ctx-ty, we can conclude that the context variable context $\Xi, \Gamma'{\uparrow}\emptyset : \mathcal{C}[\cdot]$ is well-formed. From (2) we can infer that there is a derivation for $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \cup \{\Gamma'\} \vdash F[\Gamma'/\Gamma]$ *fmla*, and from (3) it is obvious Γ' cannot appear in any formula from Ω. Therefore $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \cup \{\Gamma'\} \vdash F'$ *fmla* has a derivation of the same structure as that for $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* for all $F' \in \Omega$. But then the premise sequent $\mathbb{N}; \Psi; \Xi, \Gamma'{\uparrow}\emptyset : \mathcal{C}[\cdot]; \Omega \longrightarrow F[\Gamma'/\Gamma]$ must be well-formed, as needed.

Case: Π-L

In this case there must be a derivation of $\mathbb{N}; \Psi; \Xi \vdash \mathcal{C}[\cdot] \leadsto_{csty} G$. Given the structure of the conclusion and premise sequents, we need only show the well-formedness of the formula $F_1[G/\Gamma]$ in the premise to conclude it is a well-formed sequent as the well-formedness of the context variable context and the set of assumption formulas is assured by the well-formedness of the conclusion sequent. By the well-formedness of the conclusion sequent, $\mathbb{N} \cup \Theta \cup \Psi; \Xi^- \vdash \Pi\,\Gamma : \mathcal{C}.F_1$ *fmla* will have a derivation and thus $\mathbb{N} \cup \Theta \cup \Psi; \Xi^- \cup \{\Gamma\} \vdash F_1$ *fmla* is also derivable. The judgement $\mathbb{N}; \Psi \vdash \mathcal{C}[\cdot]$ ctx-ty has an obvious derivation, and so by Theorem 4.1 we conclude that $\mathbb{N}; \Psi \vdash G$ *context* must be derivable. Therefore by an application of Theorem 3.4 there will be a derivation of $\mathbb{N} \cup \Theta \cup \Psi; \Xi^- \vdash F_1[G/\Gamma]$ *fmla*. Thus we can conclude that the premise sequent $\mathbb{N}; \Psi; \Xi; \Omega', F_1[G/\Gamma] \longrightarrow F_2$ is well-formed.

Case: $\forall$-R

In this case the support set $\mathbb{N}$ is $\{n_1 : \alpha_1, \ldots, n_m : \alpha_m\}$ and for a new variable $y \notin dom\,(\Psi)$ of type $(\alpha_1 \rightarrow \ldots \rightarrow \alpha_m \rightarrow \alpha)$, the judgement $F[\![\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}]\!] = F'$ has a derivation. By the well-formedness of the conclusion sequent we know that

1. for each $\Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$, $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty has a derivation,

2. $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \forall x : \alpha.F$ *fmla* has a derivation, and

3. for each $F'' \in \Omega$, $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F''$ *fmla* has a derivation.

The well-formedness of the context variable context of the premise sequent is ensured by (1) and the use of Theorem 4.2, and the well-formedness of the assumption set is ensured by (3) and the use of Theorem 3.2. What remains to be shown is that the goal formula F' is well-formed. From (2) we can infer that $\mathbb{N} \cup \Theta_0 \cup \Psi, x : \alpha; \Xi^- \vdash F$ *fmla* also has a derivation. It is clear that the substitution $\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}$ will be type preserving with respect to the arity typing context $\mathbb{N} \cup \Theta_0 \cup (\Psi, y : \alpha_1 \rightarrow \ldots \rightarrow \alpha_m \rightarrow \alpha)$, and so the judgement $\mathbb{N} \cup \Theta_0 \cup \Psi, y : \alpha_1 \rightarrow \ldots \rightarrow \alpha_m \rightarrow \alpha; \Xi^- \vdash F'$ *fmla* has a derivation by Theorem 3.4. Thus the sequent $\mathbb{N}; \Psi, y : \alpha_1 \rightarrow \ldots \rightarrow \alpha_m \rightarrow \alpha; \Xi; \Omega \longrightarrow F'$ must be well-formed.

Case: $\forall$-L

In this case there is a term t such that both $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ and $F_1[\![\{\langle x, t, \alpha\rangle\}]\!] = F_1'$ have derivations. We need only show that F_1' is well-formed in the premise sequent to conclude it is a well-formed sequent, as the other formulas and the context variable types will be well-formed by the assumption that the conclusion sequent is well-formed. By the well-formedness of the conclusion sequent, there is a derivation for $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \forall x : \alpha.F_1$ *fmla* and thus for $\mathbb{N} \cup \Theta_0 \cup \Psi, x : \alpha; \Xi^- \vdash F_1$ *fmla* also. Clearly $\{\langle x, t, \alpha\rangle\}$ is arity type preserving with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ because $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ has a derivation, and so by Theorem 3.4 $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F_1'$ *fmla* has a derivation. Therefore the sequent $\mathbb{N}; \Psi; \Xi; \Omega, F_1' \longrightarrow F_2$ must be well-formed.

Case: $\exists$-R

In this case there is some term t such that both $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ and $F[\![\{\langle x, t, \alpha\rangle\}]\!] = F'$ have derivations. We need only show that the goal formula of the premise sequent is well-formed to conclude it is a well-formed sequent, as the other formulas and the context variable types will be well-formed by the assumption that the conclusion sequent is well-formed. By the well-formedness of the conclusion sequent, there exists a derivation of $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \exists x : \alpha.F$ *fmla* and thus $\mathbb{N} \cup \Theta_0 \cup \Psi, x : \alpha; \Xi^- \vdash F$ *fmla* must be derivable. Clearly $\{\langle x, t, \alpha\rangle\}$ is arity type preserving with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ because $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ has a derivation, and so by Theorem 3.4 $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* has a derivation. Therefore the sequent $\mathbb{N}; \Psi; \Xi; \Omega' \longrightarrow F'$ will be well-formed, as needed.

Case: $\exists$-L

In this case the support set $\mathbb{N}$ is $\{n_1 : \alpha_1, \ldots, n_m : \alpha_m\}$ and for a new variable $y \notin dom\,(\Psi)$ of type $\alpha_1 \to \ldots \to \alpha_m \to \alpha$, the judgement $F_1[\![\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}]\!] = F_1'$ has a derivation. By the well-formedness of the conclusion sequent we know that

1. for each $\Gamma_i {\uparrow} \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$, $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty has a derivation,

2. $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \exists x : \alpha.F_1$ *fmla* has a derivation, and

3. for each $F'' \in \Omega$, $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F''$ *fmla* has a derivation.

From (1) and the use of Theorem 4.2 we can conclude that the context variable types in Ξ will be well-formed under a context extended with y. From (2) and Theorem 3.2 we conclude the assumption formulas from Ω are well-formed under a context extended with y. Similarly, from (3) we conclude the goal formula well-formed under the extended context. It only remains to show that the formula F_1' is well-formed. From (2) we can conclude that $\mathbb{N} \cup \Theta_0 \cup \Psi, x : \alpha; \Xi^- \vdash F_1$ *fmla* also has a derivation, and since the substitution $\{\langle x, y\ n_1 \ldots n_m, \alpha\rangle\}$ is type preserving with respect to $(\Psi, y : \alpha_1 \to \ldots \to \alpha_m \to \alpha)$, $\mathbb{N} \cup \Theta_0 \cup \Psi, y : \alpha_1 \to \ldots \to \alpha_m \to \alpha; \Xi^- \vdash F_1'$ *fmla* has a derivation by Theorem 3.4. Thus it is clear that $\mathbb{N}; \Psi, y : \alpha_1 \to \ldots \to \alpha_m \to \alpha; \Xi; \Omega, F_1' \longrightarrow F_2$ is a well-formed sequent.

Case: $\wedge$-R**,** $\vee$-R_i**,** $\supset$-R**,** $\wedge$-L_i**,** $\vee$-L**, and** $\supset$-L

In all of these cases the well-formedness of the context variable types in Ξ and the formulas in Ω are ensured by the well-formedness of the conclusion sequent. In the case of the left rules we further can infer the well-formedness of the goal formulas from the well-formedness of the conclusion sequent. What remains to be shown is that the the two sub-formulas F_1

and F_2 are well-formed. By the well-formedness of the conclusion sequent there must be a derivation of $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F_1 \bullet F_2$ *fmla*. In all cases, there must then be derivations for both $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F_1$ *fmla* and $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F_2$ *fmla* by definition. Thus it is clear that all sequents appearing in the premises of these rules must be well-formed. $\square$

We recall that the validity of sequents is based on the validity of their substitution instances. In this context, the soundness of the rules Π-L, $\forall$-L and $\exists$-R depends on the ability to invert the order of application of substitutions. The three lemmas below show that this is in fact possible. Lemma 4.4 follows from an easy induction and a use of Theorem 2.3. The two following lemmas have even simpler inductive proofs.

Lemma 4.4. *Let θ_1 and θ_2 be arity type preserving substitutions with respect to Θ and $ctx(\theta_1) \uplus \Theta$ respectively and such that the variables in the domain of θ_2 are (1) distinct from the domain of θ_1 and (2) do not appear free in the range of θ_1. Then, letting*

$$\theta_2' = \{\langle x, M', \alpha\rangle \mid \langle x, M, \alpha\rangle \in \theta_2 \text{ and } M[\![\theta_1]\!] = M'\},$$

for any formula F well-formed with respect to $\text{ctx}(\theta_2) \uplus \text{ctx}(\theta_1) \uplus \Theta$ and some context variable context Ξ, $F[\![\theta_2]\!][\![\theta_1]\!] = F[\![\theta_1]\!][\![\theta_2']\!]$ has a derivation.

Lemma 4.5. *Let σ_1 and σ_2 be context variable substitutions where the context variables in the domain of σ_2 are (1) distinct from the context variables in the domain of σ_1 and (2) do not appear free in the range of σ_1. Then, letting $\sigma_2' = \{G'/\Gamma \mid G/\Gamma \in \sigma_2 \text{ and } G[\sigma_1] = G'\}$, for any formula F, $F[\sigma_2][\sigma_1] = F[\sigma_1][\sigma_2']$.*

Lemma 4.6. *Let θ be an arity type preserving substitution with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$, and let σ be a context variable substitution appropriate for Ξ with respect to $\text{ctx}(\theta) \cup \Psi$. Then, letting $\sigma' = \{G'/\Gamma \mid G/\Gamma \in \sigma \text{ and } G[\![\theta]\!] = G'\}$, for any formula F well-formed with respect to $ctx(\theta) \uplus \Psi$ and Ξ^-, $F[\sigma][\![\theta]\!] = F[\![\theta]\!][\sigma']$ is drivable.*

The soundness of the logical rules is the content of the following theorem.

Theorem 4.17 (Soundness)**.** *The following property holds for every instance of each of the rules in Figure 4.4: if the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and all the premise sequents are valid, then the conclusion sequent must also be valid.*

Proof. Consider each of the possible rules from Figure 4.4.

Case: $\top$-R

Such a sequent is always valid since $\top$ is always valid.

Case: $\bot$-L

Since $\bot$ is never valid, a sequent in which this formula appears as an assumption must obviously be valid.

Case: Π-R

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid this instance would be vacuously valid, so assume they are all valid. By the definition of substitution application, $(\Pi\,\Gamma : \mathcal{C}.F)[\![\theta]\!][\sigma] = \Pi\,\Gamma : \mathcal{C}.(F[\![\theta]\!][\sigma])$, and such a formula is valid if for every context expression G such that $\mathcal{N};\emptyset \vdash \mathcal{C} \leadsto_{cs} G$ has a derivation the formula $F[\![\theta]\!][\sigma][G/\Gamma]$ is valid. So consider an arbitrary context expression G such that $\mathcal{N};\emptyset \vdash \mathcal{C} \leadsto_{cs} G$ has a derivation. By assumption the premise sequent is $\mathbb{N};\Psi;\Xi,\Gamma' \uparrow \emptyset : \mathcal{C}[\cdot];\Omega \longrightarrow F[\Gamma'/\Gamma]$ for some $\Gamma' \notin \mathit{dom}\,(\Xi)$, and is valid. Clearly $\langle\theta,\emptyset\rangle$ and $\{G/\Gamma'\} \circ \sigma$ identify a closed instance of this sequent, and since Γ' cannot appear in any formulas in Ω all the assumption formulas of this sequent must be valid. Therefore $F'[\Gamma'/\Gamma][\![\theta]\!][\{G/\Gamma'\} \circ \sigma]$ must be a valid formula. By Lemmas 4.5 and 4.6 this formula is equivalent to $F[\![\theta]\!][\{G/\Gamma'\} \circ \sigma][G/\Gamma]$ which is further equivalent to $F[\![\theta]\!][\sigma][G/\Gamma]$ since Γ' cannot appear in F or $F[\![\theta]\!]$. From this we infer that the formula $(\Pi\,\Gamma : \mathcal{C}.F)[\![\theta]\!][\sigma]$ is valid, and thus conclude that the conclusion sequent is valid.

Case: Π-L

In this case there is a derivation for $\mathbb{N};\Psi;\Xi \vdash \mathcal{C}[\cdot] \leadsto_{csty} G$ for a context expression G. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $(\Omega, \Pi\,\Gamma : \mathcal{C}.F_1[\![\theta]\!][\sigma]$ were not valid then this instance would be vacuously valid, so suppose they are all valid. Then in particular, $(\Pi\,\Gamma : \mathcal{C}.F_1)[\![\theta]\!][\sigma]$ is valid. So for any context expression G such that $\mathcal{N}; \emptyset \vdash \mathcal{C} \rightsquigarrow_{cs} G$ is derivable, the formula $F_1[\![\theta]\!][\sigma][G/\Gamma]$ will be valid. But $G[\![\theta]\!][\sigma]$ will be such a context expression. Using Theorem 4.2 it must be that $\mathcal{N}; \Psi; \Xi \vdash \mathcal{C}[\cdot] \rightsquigarrow_{csty} G$ and so by Theorems 4.3 and 4.6 $\mathcal{N}; \emptyset; \emptyset \vdash \mathcal{C}[\cdot] \rightsquigarrow_{csty} G[\![\theta]\!][\sigma]$ will be derivable. By Theorem 4.1 we can conclude that $\mathcal{N}; \emptyset \vdash \mathcal{C} \rightsquigarrow_{cs} G[\![\theta]\!][\sigma]$, and therefore $F_1[\![\theta]\!][\sigma][G[\![\theta]\!][\sigma]/\Gamma]$ is valid. By Lemmas 4.6 and 4.5 $F_1[G/\Gamma][\![\theta]\!][\sigma] = F_1[\![\theta]\!][\sigma][G[\![\theta]\!][\sigma]/\Gamma]$ so $F_1[G/\Gamma][\![\theta]\!][\sigma]$ must be valid as well.

Thus θ and σ identify a closed instance of the premise sequent under which all the assumption formulas, $(\Omega, F_1[G/\Gamma])[\![\theta]\!][\sigma]$ are valid. But this sequent is valid by assumption and so $F_2[\![\theta]\!][\sigma]$ must be valid, and from this we can conclude that the conclusion sequent must also be valid.

Case: $\forall$-R

In this case the support set $\mathbb{N}$ is $\{n_1 : \alpha_1, \ldots, n_m : \alpha_m\}$ and for some new variable $y \notin dom\,(\Psi)$ of type $\alpha' = \alpha_1 \to \ldots \to \alpha_m \to \alpha$, $F[\![\{\langle x, y\; n_1 \ldots n_m, \alpha\rangle\}]\!] = F'$ has a derivation. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $\Omega[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. For any arbitrary term t such that $\mathcal{N} \cup \Theta_0 \vdash_{at} t : \alpha$ has a derivation there exists $t' = \lambda n_1 : \alpha_1. \ldots \lambda n_m : \alpha_m. t$ such that $\mathcal{N} \cup \Theta_0 \vdash_{at} t' : \alpha'$ ahs a derivation. Further, it is clear that $\theta' = \{\langle y, t', \alpha'\rangle\} \circ \theta$ and σ will identify a closed instance of the premise sequent in this rule. Clearly all the formulas in Ω will still be valid under these substitutions as y cannot appear in any formula in Ω, and therefore $F'[\![\theta']\!][\sigma]$ will be valid by the validity of the premise sequent. This formula is equivalent to $F'[\![\{\langle x, y\; n_1 \ldots n_m, \alpha\}]\!][\![\theta']\!][\sigma]$ which by Lemmas 4.6 and 4.4 is equivalent to $F[\![\theta']\!][\sigma][\![\{\langle x, t, \alpha\rangle\}]\!]$. Clearly y cannot appear in F and thus $F[\![\theta']\!]$ will be the same term as $F[\![\theta]\!]$. Since $F[\![\theta]\!][\sigma][\![\{\langle x, t, \alpha\rangle\}]\!]$ is valid for this arbitrary choice of term t, then $F[\![\theta]\!][\sigma]$ will be a valid formula by definition. We can therefore conclude that the conclusion sequent is valid.

Case: $\forall$-L

In this case there is a term t and derivations for $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ and $F'[\![\{\langle x, t, \alpha \rangle\}]\!] = F''$. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $(\Omega, \forall x : \alpha.F_1)[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. In particular then, $(\forall x : \alpha.F_1)[\![\theta]\!][\sigma]$ is valid. So for any term t' such that $\mathcal{N} \cup \Theta_0 \vdash_{at} t' : \alpha$ has a derivation, $F_1[\![\theta]\!][\sigma][\![\{\langle x, t', \alpha \rangle\}]\!]$ must be valid. Clearly $t' = t[\![\theta]\!]$ is such a term by Theorem 2.4, and thus $F_1[\![\theta]\!][\sigma][\![\{\langle x, t', \alpha \rangle\}]\!]$ is a valid formula. But by Lemmas 4.4 and 4.6 this must be equivalent to the formula $F_1'[\![\theta]\!][\sigma]$, and so θ and σ also identify a closed instance of the premise sequent where all of the assumption formulas are valid. Therefore $F_2[\![\theta]\!][\sigma]$ will be valid by the validity of the premise sequent, and thus the conclusion sequent must be valid.

Case: $\exists$-R

In this case there are derivations for both $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} t : \alpha$ and $F[\![\{\langle x, t, \alpha \rangle\}]\!] = F'$. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $\Omega[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. Then clearly this same θ and σ identify a closed instance of the premise sequent of this rule, and since all the formulas in Ω are valid under θ and σ we can infer from the validity of the premise sequent that $F'[\![\theta]\!][\sigma]$ must be valid. But by Lemmas 4.6 and 4.4 this formula is equal to $F[\![\theta]\!][\sigma][\![\{\langle x, t[\![\theta]\!], \alpha \rangle\}]\!]$ and there is a derivation of $\mathcal{N} \cup \Theta_0 \vdash_{at} t[\![\theta]\!] : \alpha$ by Theorem 2.4. Thus $\exists x : \alpha.F[\![\theta]\!][\sigma]$ must be valid. Therefore the conclusion sequent will be valid.

Case: $\exists$-L

In this case the support set $\mathbb{N}$ is $\{n_1 : \alpha_1, \ldots, n_m : \alpha_m\}$ and for a new variable $y \notin dom(\Psi)$ of type $\alpha' = (\alpha_1 \to \ldots \to \alpha_m \to \alpha)$, the judgement $F_1[\![\{\langle x, y\ n_1 \ldots n_m, \alpha \rangle\}]\!] = F_1'$ has a derivation. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $(\Omega, \exists x : \alpha.F_1)[\![\theta]\!][\sigma]$ were not valid this closed instance would be vacuously valid, so assume all these formulas are valid. Then in particular, $(\exists x : \alpha.F_1)[\![\theta]\!][\sigma]$ will be valid. So there must exist some term t where $\mathcal{N} \cup \Theta_0 \vdash_{at} t : \alpha$ has a derivation and $F_1[\![\theta]\!][\sigma][\![\{\langle x, t, \alpha \rangle\}]\!]$ is valid. From this we construct a term $t' = \lambda n_1 : \alpha_1. \ldots \lambda n_m : \alpha_m. t$

which is such that $\{\langle y, t', \alpha' \rangle\} \circ \theta$ and σ identify a closed instance of the premise sequent. By Lemmas 4.4 and 4.6 the equality $F_1'[\![\{\langle y, t', \alpha' \rangle\} \circ \theta]\!][\sigma] = F_1[\![\{\langle y, t', \alpha' \rangle\} \circ \theta]\!][\sigma][\![\{\langle x, t, \alpha \rangle\}]\!]$ holds and as y cannot have appeared in F_1 this is also equivalent to $F_1[\![\theta]\!][\sigma][\![\{\langle x, t, \alpha \rangle\}]\!]$ which we know to be valid. Similarly all formulas in $\Omega[\![\{\langle y, t', \alpha' \rangle\} \circ \theta]\!][\sigma]$ will be valid as y cannot appear in any formula in Ω and all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid.

Thus by the validity of the premise sequent, $F_1[\![\{\langle y, t', \alpha' \rangle\} \circ \theta]\!][\sigma]$ must be valid, and again because y cannot have appeared in F_1 the formula $(\exists x : \alpha.F_1)[\![\theta]\!][\sigma]$ is valid. Therefore we can conclude that the conclusion sequent will be valid.

Case: $\wedge$-R

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $\Omega[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. There are two premise sequents in this rule: $\mathcal{S}_1 = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_1$ and $\mathcal{S}_2 = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_2$. Clearly θ and σ also identify a closed instance for both $\mathcal{S}_1$ and $\mathcal{S}_2$, and since all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid both $F_1[\![\theta]\!][\sigma]$ and $F_2[\![\theta]\!][\sigma]$ must be valid by the validity of these premise sequents. But then clearly $(F_1 \wedge F_2)[\![\theta]\!][\sigma]$ is valid by Definition 3.2.

Case: $\wedge$-L_i

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $(\Omega, F_1 \wedge F_2)[\![\theta]\!][\sigma]$ were not valid this closed instance would be vacuously valid, so assume all these formulas are valid. Then in particular, $(F_1 \wedge F_2)[\![\theta]\!][\sigma]$ will be valid and so by definition $F_1[\![\theta]\!][\sigma]$ and $F_2[\![\theta]\!][\sigma]$ are both valid.

The premise sequent is of the form $\mathbb{N}; \Psi; \Xi; \Omega', F_i \longrightarrow F$ for some $i \in \{1, 2\}$. Clearly θ and σ identify a closed instance of this sequent, and as all formulas in $(\Omega, F_i)[\![\theta]\!][\sigma]$ are valid, the formula $F_i[\![\theta]\!][\sigma]$ must be valid. So by the validity of the premise sequent, $F[\![\theta]\!][\sigma]$ is valid, and thus we can conclude that the conclusion sequent is valid.

Case: $\vee$-R_i

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $\Omega[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. There is a single premise sequent in this rule and it is of the form $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_i$ for some $i \in \{1, 2\}$. As θ and σ clearly identify a closed instance of this premise sequent, by the validity of this sequent we can conclude that $F_i[\![\theta]\!][\sigma]$ is valid. So by Definition 3.2 the formula $(F_1 \vee F_2)[\![\theta]\!][\sigma]$ must be valid, and therefore the conclusion sequent is valid.

Case: $\vee$-L

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $(\Omega, F_1 \vee F_2)[\![\theta]\!][\sigma]$ were not valid this closed instance would be vacuously valid, so assume all these formulas are valid. Then in particular, $(F_1 \vee F_2)[\![\theta]\!][\sigma]$ will be valid.

There are two premise sequents of this rule: $\mathcal{S}_1 = \mathbb{N}; \Psi; \Xi; \Omega, F_1 \longrightarrow F$ and $\mathcal{S}_2 = \mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F$. Note that θ and σ identify closed instances of both these sequents. Since $(F_1 \vee F_2)[\![\theta]\!][\sigma]$ is valid, then either $F_1[\![\theta]\!][\sigma]$ is valid or $F_2[\![\theta]\!][\sigma]$ is valid. In the former case, we can infer from the validity of $\mathcal{S}_1$ that $F[\![\theta]\!][\sigma]$ must be valid, and in the later case we infer the same but through the validity of $\mathcal{S}_2$. Therefore $F[\![\theta]\!][\sigma]$ will always be valid, and we can conclude that the conclusion sequent is valid.

Case: $\supset$-R

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formulas in $\Omega[\![\theta]\!][\sigma]$ were not valid then this closed instance would be vacuously valid, so assume that they are all valid. There is a single premise sequent in this rule, $\mathbb{N}; \Psi; \Xi; \Omega, F_1 \longrightarrow F_2$, and θ and σ clearly identify a closed instance of this sequent. Whenever the formula $F_1[\![\theta]\!][\sigma]$ is valid we also know that all the formulas in $(\Omega, F_1)[\![\theta]\!][\sigma]$ are valid and so by the validity of this premise sequent we can infer that $F_2[\![\theta]\!][\sigma]$ is valid. But then $F_1[\![\theta]\!][\sigma] \supset F_1[\![\theta]\!][\sigma]$ must be valid by the semantics of formulas. This is the same as $(F_1 \supset F_2)[\![\theta]\!][\sigma]$ by the definition of substitution applications, and therefore we conclude that the conclusion sequent is valid.

Case: $\supset$-L

Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $(\Omega, F_1 \supset F_2)[\![\theta]\!][\sigma]$ were not valid this closed instance would be vacuously valid, so assume all these formulas are valid. Then in particular, $(F_1 \supset F_2)[\![\theta]\!][\sigma]$ is valid. There are two premise sequents of this rule which are valid by assumption: $\mathcal{S}_1 = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_1$ and $\mathcal{S}_2 = \mathbb{N}; \Psi; \Xi; \Omega, F_2 \longrightarrow F$. Both $\mathcal{S}_1$ and $\mathcal{S}_2$ have closed instances identified by θ and σ as they share arity and context variables contexts with the conclusion sequent. By assumption all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid, and so by the validity of $\mathcal{S}_1$ the formula $F_1[\![\theta]\!][\sigma]$ is valid. Thus, by the validity of $(F_1 \supset F_2)[\![\theta]\!][\sigma]$, the formula $F_2[\![\theta]\!][\sigma]$ must be valid. But then all the formulas in $(\Omega, F_2)[\![\theta]\!][\sigma]$ are valid, and so by the validity of $\mathcal{S}_2$ we can conclude the formula $F[\![\theta]\!][\sigma]$ is valid. We can therefore conclude that the conclusion sequent is valid. □

4.3 Proof Rules that Interpret Atomic Formulas

Atomic formulas in our logic and sequents represent LF typing judgements. The validity of such formulas is therefore determined by the LF derivations rules and this fact can be used in their analysis. When an atomic formula appears as the conclusion of a sequent, the analysis takes an obvious form: the derivability of the sequent can be based on that of a sequent in which the conclusion judgement has been unfolded using an LF rule. The treatment of an atomic assumption formula requires more thought. Such formulas may contain context and eigenvariables in them and we must consider all the possible instantiations for these variables that could make the judgement true in determining the validity of the sequent. The translation of this general requirement into an analysis that is local to the atomic formula may be driven by the structure of the type in the formula. When the type is of the form $\Pi x{:}A.\, B$, the term must be an abstraction and there is exactly one way in which a purported typing derivation could have concluded. When the type is atomic, Theorem 2.12 provides us information about the different cases that need to be considered.

We transform the ideas outlined above into a concrete collection of proof rules in this section. The development of a "case analysis rule" for atomic assumption formulas is somewhat intricate and is the subject of the first subsection below. With this done, the second subsection presents the proof rules and shows that they obey the important properties of soundness and preservation of wellformedness for sequents.

4.3.1 Analyzing an Atomic Assumption Formula with an Atomic Type

Our goal here is to develop the basis for a proof rule around the analysis of an assumption of the form $\{G \vdash R : P\}$. Obviously P in this case is of the form $(a\ M_1\ \ldots\ M_n)$. Let us suppose initially that this formula is closed. In this case, for the formula to be valid, R would need to have as a head a constant declared in Σ or a nominal constant assigned a type in G. If the arguments of R do not satisfy the constraints imposed by the type associated with the head, then the typing judgement will not be derivable and hence we can conclude that the sequent is in fact valid. On the other hand, if the arguments of R do satisfy the required constraints, then Theorem 2.12 gives us a means for decomposing the given typing judgement into ones pertaining to $M_1, \ldots, M_n$. The validity of the given sequent can therefore be reduced to the validity of a sequent that results from replacing the atomic formula under consideration by ones that represent the mentioned typing judgements. In this discussion, we have implicitly assumed that G is well-formed. However, the reduction described is easily seen to be sound even when G is not well-formed.

In the general case, the formula $\{G \vdash R : P\}$ may not be closed. There are, in fact, two conceptually different possibilities that need to be considered from this perspective. First, the context expression may have a part that is yet to be determined, i.e., G may be of the form $\Gamma, n_1 : A_1, \ldots, n_m : A_m$ where Γ has a set of names $\mathbb{N}$ and a context variable type of the form $\mathcal{C}[G_1; \ldots; G_\ell]$ associated with it in the sequent. Second, the expressions in the atomic formula and the context variable type may contain variables in them that are bound in the eigenvariable context. To articulate a proof rule around the atomic formula in this situation, it is necessary to develop a means for analyzing the formula in a way that pays attention to the validity of the sequent under all acceptable instantiations of the context and term variables.

The analysis that we describe proceeds in two steps. We first describe a finite way to consider elaborations of the context variable that make explicit all the heads that need to be considered for the term in an analysis of the closed instances of the atomic formula. Concretely, this process yields a finite collection of pairs comprising a sequent in which the context variable may have been partially instantiated, and a specifically identified possibility for the head that is either drawn from the signature or that appears explicitly in the context; the intent here, which is verified in Lemma 4.8, is that considering just the second components of these pairs as the heads of the term in the typing judgement will suffice for a complete analysis based on Theorem 2.12. The second step actually carries out the analysis in each of these cases, using the idea of unification in the application of Theorem 2.12 to accommodate all possible closed instantiations of the term variables in the sequent. We consider these steps in the two subsections below.

Elaborating Context Variables and Identifying Head Possibilities

We first note that context expressions have may have implicit and explicit parts, the former being subject to elaboration via context substitutions.

Definition 4.11 (Implicit and Explicit Parts of a Context)**.** Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ be a well-formed sequent. If G is a context expression appearing in $\mathcal{S}$ then it must be of either the form $n_1 : A_1, \ldots, n_m : A_m$ or of the form $\Gamma, n_1 : A_1, \ldots, n_m : A_m$ where Γ is a context variable with an associated declaration $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n]$ in Ξ. In the latter case, we say that G has an implicit part relative to $\mathcal{S}$ that is given by $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n]$. Further, we refer to $n_1 : A_1, \ldots, n_m : A_m$ in the former case and to the sequence formed by listing the bindings in $G_1, \ldots, G_n$ followed by $n_1 : A_1, \ldots, n_m : A_m$ in the latter case as the explicit bindings in G relative to $\mathcal{S}$.

Let Γ be a context variable that has the type $\mathcal{C}[G_1; \ldots; G_\ell]$. Closed instances of Γ are then generated by interspersing $G_1, \ldots, G_\ell$ with blocks of declarations generated from the block schema comprising $\mathcal{C}$. In determining possibilities for the head of R from the implicit part of G in an atomic formula of the form $\{G \vdash R : P\}$, we need to consider an elaboration of G with only one such block; of course, for a complete analysis, we will need to consider all the possibilities for such an elaboration. The function *AddBlock* defined below formalizes such an elaboration, returning a modified sequent and a potential head for the term in the typing judgement. Note that in an elaboration based on a block schema of the form $\{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}y_1 : A_1, \ldots, y_k : A_k$, it would be necessary to consider a choice of nominal constants for the schematic variables $y_1, \ldots, y_k$. The function is parameterized by such a choice. We must also accommodate all possible instantiations for the variables $x_1, \ldots, x_n$, subject to the proviso that these instantiations do not use nominal constants that appear in a later part of the context expression. This is done by using (implicitly universally quantified) term variables for $x_1, \ldots, x_n$ and by raising such variables over the nominal constants that are not prohibited from appearing in the instantiations; to support the latter requirement, the function is parameterized by a collection of nominal constants. Finally, we observe that the elaboration process may introduce new nominal constants into the sequent, necessitating a raising of the eigenvariables over the new constants.

Definition 4.12 (Adding a Block and Picking a Binding From It)**.** Let $\mathcal{S}$ be the the well-formed sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$, let there be some $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n] \in \Xi$, and let $\mathcal{B} = \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}y_1 : A_1, \ldots, y_k : A_k$ be one of the block schemas comprising $\mathcal{C}$. Further, let $\mathbb{N}' \subseteq (\mathbb{N} \setminus \mathbb{N}_\Gamma)$ be a collection of nominal constants, let ns be a list $n_1, \ldots, n_k$ of distinct nominal constants that are also different from the constants in $\mathbb{N}'$ and that are such that, for $1 \leq i \leq k$, $n_i : (A_i)^- \in (\mathcal{N} \setminus \mathbb{N}_\Gamma)$. Finally, for $0 \leq j \leq n$, let $\mathbb{N}_j$ be the collection of nominal constants assigned types in $G_1, \ldots, G_j$. Then, letting

1. Ψ'_j be a version of Ψ raised over $\{n_1, \ldots, n_k\} \setminus \mathbb{N}$, θ'_j be the associated raising substitution,

2. $A'_1, \ldots, A'_k$ be the types $A_1, \ldots, A_k$ with the schematic variables $y_1, \ldots, y_k$ replaced with the names $n_1, \ldots, n_k$,

3. Ψ''_j be a version of $\{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$ raised over $\mathbb{N}' \cup \mathbb{N}_j \cup (\{n_1, \ldots, n_k\} \setminus \mathbb{N})$ with the new variables chosen to be distinct from those in Ψ'_j, θ''_j be the associated raising substitution, G be the context expression $n_1 : A'_1[\![\theta''_j]\!], \ldots, n_k : A'_k[\![\theta''_j]\!]$, and

4. Ξ'_j be the context variable context

$$(\Xi \setminus \{\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n]\}) \llbracket \theta'_j \rrbracket$$
$$\bigcup$$
$$\{\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1\llbracket \theta'_j \rrbracket; \ldots; G_j\llbracket \theta'_j \rrbracket; G; G_{j+1}\llbracket \theta'_j \rrbracket; \ldots; G_n\llbracket \theta'_j \rrbracket]\},$$

for $0 \leq j \leq n$ and $1 \leq i \leq k$, $\mathit{AddBlock}(\mathcal{S}, \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n], \mathcal{B}, ns, \mathbb{N}', j, i)$ is defined to be the tuple

$$\langle \mathbb{N} \cup ns; \Psi'_j \cup \Psi''_j; \Xi'_j; \Omega\llbracket \theta'_j \rrbracket \longrightarrow F\llbracket \theta'_j \rrbracket, n_i : A'_i\llbracket \theta''_j \rrbracket \rangle.$$

Note that the conditions in the definition ensure that all the substitutions involved in it will have a result, thereby permitting us to use the notation introduced after Theorem 2.4.

The elaboration just described is parameterized by the choice of nominal constants for the variables assigned types in the block schema. In identifying the choices that have to be considered, it is useful to partition the members of $(\mathcal{N} \setminus \mathbb{N}_\Gamma)$ into two sets: those that appear in the support set of the sequent whose elaboration is being considered and those that do not. It is necessary to consider all possible assignments that satisfy arity typing constraints from the first category. From the second category, as we shall soon see, it suffices to consider exactly one representative assignment. Note also that we may insist that the nominal constant in each assignment of the block be disinct; if this is not the case, the sequent is easily seen to be valid. The function *NamesLsts* defined below embodies these ideas. The function is parameterized be a sequence of arity types corresponding to the declarations in the block schema, a collection of "known" nominal constants that are available for use in an elaboration of the block schema and a collection of nominal constants that are already bound in the context expressions and hence must not be used again.

Definition 4.13 (Identifying a Choice of Nominal Constants)**.** Let tys be a sequence of arity types and let $\mathbb{N}_o$ and $\mathbb{N}_b$ be finite sets of nominal constants. Further, let *nil* denote an empty sequence and $x :: xs$ denote a sequence that starts with x and continues with the sequence xs. Then the collection of name choices for tys relative to $\mathbb{N}_o$ and away from $\mathbb{N}_b$ is denoted by $\mathit{NamesLsts}(tys, \mathbb{N}_o, \mathbb{N}_b)$ and defined by recursion on tys as follows:

$$\mathit{NamesLsts}(tys, \mathbb{N}_o, \mathbb{N}_b)) =
\begin{cases}
\{\mathit{nil}\} & \text{if } tys = \mathit{nil} \\[2ex]
\{n :: nl \mid n : \alpha \in \mathcal{N}, n \in \mathbb{N}_o \setminus \mathbb{N}_b, \text{ and} & \\
\qquad nl \in \mathit{NamesLsts}(tys', \mathbb{N}_o, \mathbb{N}_b \cup \{n\})\} \cup & \\
\{n :: nl \mid n \text{ is the first nominal constant} & \text{if } tys = \alpha :: tys' \\
\qquad \text{such that } n : \alpha \in \mathcal{N} \text{ and } n \notin \mathbb{N}_o \cup \mathbb{N}_b, & \\
\qquad \text{and } nl \in \mathit{NamesLsts}(tys', \mathbb{N}_o, \mathbb{N}_b \cup \{n\})\} &
\end{cases}$$

We assume in this definition the existence of an ordering on the nominal constants that allows us to select the first of these constants that satisfies a criterion of interest.

We can now identify a finite collection of elaborations of the implicit part of a context expression that must be considered in the analysis of an assumption formula of the form $\{G \vdash R : P\}$ that appears in a sequent $\mathcal{S}$. We do this below through the definition of the function *ImplicitHeads*.

Definition 4.14 (Head Choices from the Implicit Part of a Context)**.** Let $\mathcal{S}$ be a well-formed sequent $\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$, let G be a context expression appearing in a formula in $\mathcal{S}$ that has an implicit part relative to $\mathcal{S}$ that is given by $\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n]$, and let $\mathcal{B} = \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\} y_1 : A_1, \ldots, y_k : \alpha_k$ be one of the block schemas comprising $\mathcal{C}$. Further, let $\mathbb{N}_b$ be the collection of nominal constants assigned types by the explicit bindings of G relative to $\mathcal{S}$ and let $\mathbb{N}_o = \mathbb{N} \setminus \mathbb{N}_\Gamma \setminus \mathbb{N}_b$. Finally, let $\mathit{AllBlocks}(\mathcal{S}, \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n], \mathcal{B})$ denote the set

$$\{\mathit{AddBlock}(\mathcal{S}, \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n], \mathcal{B}, ns, \mathbb{N}_o, j, i) \mid \\ 0 \leq j \leq n, 1 \leq i \leq k, ns \in \mathit{NamesLsts}(((A_1)^-, \ldots, (A_k)^-), \mathbb{N}_o, \mathbb{N}_\Gamma \cup \mathbb{N}_b)\}.$$

If $\{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ is the collection of of block schemas comprising $\mathcal{C}$, then the implicit heads in G relative to $\mathcal{S}$ is defined to be the set

$$\bigcup \{\mathit{AllBlocks}(\mathcal{S}, \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_n], \mathcal{B}) \mid \mathcal{B} \in \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}\}.$$

This set is denoted by $\mathit{ImplicitHeads}(\mathcal{S}, G)$.

The complete set of heads and corresponding (elaborated) sequents that must be considered in the analysis of an atomic formula of the form $\{G \vdash R : P\}$ is identified through the function *Heads* that is defined below.

Definition 4.15 (The Complete Set of Head Choices)**.** Let $\mathcal{S}$ be a well-formed sequent and let G be a context expression appearing in a formula in $\mathcal{S}$. Let *NewHds* be the set $\mathit{ImplicitHeads}(\mathcal{S}, G)$ if G has an implicit part relative to $\mathcal{S}$ and the empty set otherwise. Then the heads in G relative to $\mathcal{S}$ is defined to be the set

$$\{\langle \mathcal{S}, c : A\rangle \mid c : A \in \Sigma\} \cup \{\langle \mathcal{S}, n : A\rangle \mid n : A \text{ is an explicit binding in G relative to } \mathcal{S}\}$$
$$\bigcup$$
$$\mathit{NewHds}.$$

This set is denoted by $\mathit{Heads}(\mathcal{S}, G)$.

The first property that we observe of the elaboration process described is that it requires us to consider only well-formed sequents.

Lemma 4.7. *Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ be a well-formed sequent and let $\{G \vdash R : P\}$ be an atomic formula in Ω. Then for each $(\mathcal{S}', h : A) \in \mathit{Heads}(\mathcal{S}, G)$ it must be the case that $\mathcal{S}'$ is a well-formed sequent. Further, if $\mathcal{S}'$ is $\mathbb{N}'; \Psi'; \Xi'; \Omega' \longrightarrow F'$, it must be the case that $(\mathbb{N}' \cup \Theta_0 \cup \Psi') \vdash_{ak} A$ type is derivable.*

Proof. The claim is not immediately obvious only when $(\mathcal{S}', h : A) \in \mathit{ImplicitHeads}(\mathcal{S}, G)$. For these cases, it suffices to show that every pair generated by *AddBlock* satisfies the requirements of the lemma. However, this is easily argued. The main observation–that gets used twice—is that if Ψ_2 is a version of Ψ_1 raised over some collection of nominal constants $\mathbb{N}_2$ with θ being the associated raising substitution, and $\Theta \cup \Psi_1 \vdash_{ak} A'$ *type* holds for some arity context Θ that is disjoint from Ψ_1 and Ψ_2, then $\Theta \cup \Psi_2 \cup \mathbb{N}_2 \vdash_{ak} A'[\![\theta]\!]$ *type* also holds. $\square$

We want next to show the adequacy of the elaboration process, i.e., that the collection of pairs of sequents and heads it identifies are sufficient for the analysis of validity for a sequent with an assumption formula of the form $\{G \vdash R : P\}$. One aspect that we must account for in our argument is that we consider all possible choices for a "new name" for a binding in a block instance through a single representative. The key property that enables this reduction is that the validity of closed sequents is invariant under permutations of the nominal constants as we discussed in Theorem 4.11.

The following lemma in combination with Theorem 4.11 yields the desired result concerning the elaboration process.

Lemma 4.8. *Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F$ be a well-formed sequent and let $\{G \vdash R : P\}$ be a formula in Ω. Further, let θ and σ be term and context variable substitutions that identify a closed instance of $\mathcal{S}$ and that are such that $\{G \vdash R : P\}\,[\![\theta]\!][\sigma]$ is valid. If the term $R[\![\theta]\!] = (h\ M_1 \ldots M_n)$, then there is a pair $\langle \mathcal{S}', h' : A'\rangle$ in Heads$(\mathcal{S}, G)$ such that*

1. *there is a formula $\{G' \vdash R' : P'\}$ amongst the assumption formulas of $\mathcal{S}'$ with $h' : A'$ appearing in either Σ or in the explicit bindings in G' relative to $\mathcal{S}'$, and*

2. *there is a closed instance of $\mathcal{S}'$ identified by closed term and context variable substitutions θ' and σ' and a permutation π such that $\pi.h' = h$, $\pi.\mathcal{S}'[\![\theta']\!]_{\emptyset}[\sigma'] = \mathcal{S}[\![\theta]\!]_{\emptyset}[\sigma]$, and $\pi.\{G' \vdash R' : P'\}\,[\![\theta']\!][\sigma'] = \{G \vdash R : P\}\,[\![\theta]\!][\sigma]$.*

Proof. Since $\{G \vdash R : P\}\,[\![\theta]\!][\sigma]$ is valid, it must be the case that there are LF derivations for $\vdash_\Sigma G[\![\theta]\!][\sigma]$ $\texttt{ctx}$, $G[\![\theta]\!][\sigma] \vdash_\Sigma P[\![\theta]\!]$ $\texttt{type}$, and $G[\![\theta]\!][\sigma] \vdash_\Sigma R[\![\theta]\!] \Leftarrow P[\![\theta]\!]$. Using Theorem 2.12 together with the fact that $R[\![\theta]\!] = (h\ M_1 \ldots M_n)$, we see that, for an appropriate A, $h : A$ must be a member of Σ or it must appear in $G[\![\theta]\!][\sigma]$. Our argument distinguishes two ways that this could happen: it could be because $h : A$ is a member of Σ or it is an instance of a declaration in the explicit part of G or because it is introduced into the context $G[\![\theta]\!][\sigma]$ by the substitution σ.

The first collection of cases is easily dealt with: essentially, we pick h', $\mathcal{S}'$, θ' and σ' to be identical to h, $\mathcal{S}$, θ and σ, respectively, and we let π be the identity permutation. The requirements of the lemma then follow easily from the definition of the *Heads* function.

In the cases that remain, G must have the form $\Gamma, n_1^G : A_1^G, \ldots, n_p^G : A_p^G$ for some context variable Γ that has the set of names $\mathbb{N}_\Gamma$ and the type $\mathcal{C}[G_1; \ldots; G_\ell]$ assigned to it in Ξ and h must be introduced by the substitution that σ makes for Γ as the i^{th} binding, for some i, in a block of declarations resulting from instantiating one of the block schemas constituting $\mathcal{C}$. Let us suppose the relevant block schema is $\mathcal{B}$ and it has the form $\{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}(y_1 : B_1, \ldots, y_k : B_k)$. Moreover, let us suppose that this block of declarations appears in $G[\![\theta]\!][\sigma]$ somewhere between the instances of G_j and G_{j+1}, for some j between 0 and ℓ. We may, without loss of generality, assume $x_1, \ldots, x_n$ to be distinct from the variables assigned types by Ψ. We can then visualize the block introducing h as $(n_1 : B_1'[\![\theta^h]\!], \ldots, n_k : B_k'[\![\theta^h]\!])$ for some $n_1, \ldots, n_k$ of the requisite types, for some types $B_1', \ldots, B_k'$ which are the types $B_1, \ldots, B_k$ with the schematic variables of the schema replaced by these names, and for a closed substitution θ^h whose domain is $x_1, \ldots, x_n$ and, since $\vdash_\Sigma G[\![\theta]\!][\sigma]\ \mathtt{ctx}$ is derivable, whose support does not contain the nominal constants in $\mathbb{N}_\Gamma$, $n_1^G, \ldots, n_p^G$, or those that are assigned a type in $G_{j+1}, \ldots, G_\ell$. It follows from this that if we can associate the type $\mathcal{C}[G_1[\![\theta]\!]; \ldots; G_j[\![\theta]\!]; n_1 : B_1'[\![\theta^h]\!], \ldots, n_k : B_k'[\![\theta^h]\!]; G_{j+1}[\![\theta]\!]; \ldots; G_\ell[\![\theta]\!]]$ with Γ, then the context expression that σ substitutes for Γ can still be generated from the changed type. The key to our showing that the requirements of the lemma are met in these cases will be to establish that *Heads*$(\mathcal{S}, G)$ contains a sequent and head pair such that the type of Γ is elaborated to a form from which the above type can be obtained, up to a permutation of nominal constants, by a well-behaved substitution and the head is identified as the i^{th} item in the introduced block of declarations.

Towards this end, let us consider the tuple $\langle \mathcal{S}'', h'' : A'' \rangle$ that is generated by

$$AddBlock(\mathcal{S}, \Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_\ell], \mathcal{B}, (n_1, \ldots, n_k), \mathbb{N}_o, j, i),$$

where $\mathbb{N}_o$ is the collection of nominal constants obtained by leaving out of $\mathbb{N}$ the constants in $\mathbb{N}_\Gamma$ and the constants that appear amongst the explicit bindings of G relative to $\mathcal{S}$. In this case, $\mathcal{S}''$ will have the form $\mathbb{N}''; \Psi''; \Xi''; \Omega'' \longrightarrow F''$ with the following properties. First, $\mathbb{N}''$ will be identical to $\mathbb{N} \cup \{n_1, \ldots, n_k\}$. Second, Ψ'' will comprise two disjoint parts $\Psi_r^{\mathcal{S}}$ and $\Psi_r^{\mathcal{B}}$, where $\Psi_r^{\mathcal{S}}$ is a version of Ψ raised over the nominal constants in $\{n_1, \ldots, n_k\}$ that are not members of $\mathbb{N}$ with a corresponding raising substitution θ_r^{Ψ}, and $\Psi_r^{\mathcal{B}}$ is a version of $\{x_1, \ldots, x_n\}$ raised over all the nominal constants in $\mathbb{N} \cup \{n_1, \ldots, n_k\}$ except the ones that are assigned a type in $G_{j+1}, \ldots, G_\ell$ or that appear in $n_1^G, \ldots, n_p^G$ with the corresponding raising substitution $\theta_r^{\mathcal{B}}$. Third, Ξ'' will be

$$\hat{\Xi} \cup \{\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1[\![\theta_r^{\Psi}]\!]; \ldots, G_j[\![\theta_r^{\Psi}]\!]; n_1 : B_1'[\![\theta_r^{\mathcal{B}}]\!], \ldots, n_k : B_k'[\![\theta_r^{\mathcal{B}}]\!]; G_{j+1}[\![\theta_r^{\Psi}]\!], \ldots G_\ell[\![\theta_r^{\Psi}]\!]]\}$$

where $\hat{\Xi} = (\Xi \setminus \{\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_\ell]\})[\![\theta_r^{\Psi}]\!]$. Finally, each formula in $\Omega'' \cup \{F''\}$ is obtained by applying the raising substitution θ_r^{Ψ} to a corresponding one in $\Omega \cup \{F\}$. Using Theorem 4.4 we observe that, because $supp(\theta)$ is disjoint from the set $\mathbb{N}$, there is a (closed) raising substitution θ_r with $ctx(\theta_r) = \Psi_r^{\mathcal{S}}$ whose support is disjoint from the set $\mathbb{N} \cup \{n_1, \ldots, n_k\}$ and which is such that $\Omega''[\![\theta_r]\!] = \Omega[\![\theta]\!]$, $F''[\![\theta_r]\!] = F[\![\theta]\!]$, $\hat{\Xi}[\![\theta_r]\!]$ is equal to $(\Xi \setminus \{\Gamma \uparrow \mathbb{N}_\Gamma : \mathcal{C}[G_1; \ldots; G_\ell]\})[\![\theta]\!]$, and, for each q, $1 \leq q \leq \ell$, $G_q[\![\theta_r^{\Psi}]\!][\![\theta_r]\!] = G_q[\![\theta]\!]$. Using Theorem 4.4 again, we see that there is a (closed) substitution θ_r^h with $ctx(\theta_r^h) = \Psi_r^{\mathcal{B}}$ whose support is disjoint from $\mathbb{N} \cup \{n_1, \ldots, n_k\}$ and that is such that, for $1 \leq q \leq k$, it is the case that $B_q'[\![\theta_q^{\mathcal{B}}]\!][\![\theta_r^h]\!] = B_q'[\![\theta^h]\!]$. Based on all these observations, it is easy to see that if we let $\theta'' = \theta_r \cup \theta_r^h$, then $\langle \theta'', \emptyset \rangle$ is substitution compatible with $\mathcal{S}''$ and $\mathcal{S}''[\![\theta'']\!]_\emptyset$ is identical to $\mathcal{S}[\![\theta]\!]_\emptyset$ except for the fact that the type associated with Γ in its context variable context is $\mathcal{C}[G_1[\![\theta]\!]; \ldots; G_j[\![\theta]\!]; n_1 : B_1'[\![\theta^h]\!], \ldots, n_k : B_k'[\![\theta^h]\!]; G_{j+1}[\![\theta]\!]; \ldots; G_\ell[\![\theta]\!]]$. By the earlier observation, σ is appropriate for $\mathcal{S}''[\![\theta'']\!]_\emptyset$ and, in fact $\mathcal{S}''[\![\theta'']\!]_\emptyset[\sigma] = \mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$. Noting also that $h'' : A''$ must, by the definition of $AddBlock$, be $n_i : B_i'[\![\theta_r^{\mathcal{B}}]\!]$, if $Heads(\mathcal{S}, G)$ includes in it a pair obtained by this particular call to $AddBlock$, then we can pick $\mathcal{S}'$ to be $\mathcal{S}''$, h' to be

h'', A' to be A'', θ' to be θ'', σ' to be σ and π to be the identity permutation to satisfy the requirements of the lemma.

We are, of course, not assured that there will be a pair in $\mathit{Heads}(\mathcal{S}, G)$ corresponding to the use of *AddBlock* with exactly the arguments considered above. Specifically, the sequences of nominal constants that are considered for the block instance may not include $n_1, \ldots, n_k$. However, we know that some sequence $n'_1, \ldots, n'_k$ will be considered that is identical to $n_1, \ldots, n_k$ except for constants in identical locations in the two sequences that are not drawn from $\mathbb{N}$. Since the constants in any sequence must be distinct, it follows easily that we can describe a permutation π' on the nominal constants that is the identity map on $\mathbb{N}$ and that maps $n'_1, \ldots, n'_k$ to $n_1, \ldots, n_k$. It can also be seen then that $\mathit{Heads}(\mathcal{S}, G)$ will include a tuple $\langle \mathcal{S}''', h''' : A''' \rangle$ such that $\pi'.\mathcal{S}''' = \mathcal{S}''$, $\pi'.h''' = h''$, and $\pi'.A''' = A''$. Picking $\mathcal{S}'$ to be $\mathcal{S}'''$, h' to be h''', A' to be A''', θ' to be ${\pi'}^{-1}.\theta''$, σ' to be ${\pi'}^{-1}.\sigma''$, π to be π' and using Theorems 4.9 and 4.10, we can once again see that the requirements of the lemma are met. $\square$

Generating a Covering Set of Premise Sequents

Given a sequent $\mathcal{S}$ and a particular atomic assumption formula F with context expression G in $\mathcal{S}$, Lemma 4.8 assures us that $\mathit{Heads}(\mathcal{S}, G)$ correctly identifies all the context elaborations and corresponding heads that need to be considered in the analysis of F. However, we are still left with the task of identifying a systematic way of considering all the term and context substitutions that yield closed instances of $\mathcal{S}$ in which the term component of F has the relevant head. We now turn to this task. Rather than identifying the closed instances immediately, we will think of taking a step in this direction that also allows us to reduce the typing judgement represented by F based on the typing rule for the LF judgement it represents; this analysis will then be reflected in a proof rule in our system. The first step in this direction will be to determine a substitution that makes the head of F identical to the one it needs to be in its closed form. We use the idea of unification, refined to fit our context, towards this end. We describe next the idea of reducing a sequent that encodes the analysis of an LF typing judgement based on the observations in Theorem 2.12. The eventual proof rule will then combine the identification of relevant heads using $\mathit{Heads}(\mathcal{S}, G)$, the solving of a unification problem based on each such head, and the reduction of the sequent.

We begin this development by first elaborating the notion of unification problems and their solutions.

Definition 4.16 (Unification Problems & their Solutions)**.** A *unification problem* $\mathcal{U}$ is a tuple $\langle \mathbb{N}; \Psi; \mathcal{E} \rangle$ in which $\mathbb{N}$ is a collection of nominal constants, Ψ is a collection of arity type assignments to term variables, and $\mathcal{E}$ is a set of the form $\{E_1 = E'_1, \ldots, E_n = E'_n\}$ where, for each i, $1 \leq i \leq n$, either $\mathbb{N} \cup \Psi \cup \Theta_0 \vdash_{ak} E_i \; \mathit{type}$ and $\mathbb{N} \cup \Psi \cup \Theta_0 \vdash_{ak} E'_i \; \mathit{type}$ have derivations or there is an arity type α such that $\mathbb{N} \cup \Psi \cup \Theta_0 \vdash_{at} E_i : \alpha$ and $\mathbb{N} \cup \Psi \cup \Theta_0 \vdash_{at} E'_i : \alpha$ have derivations. A *solution* to the unification problem $\mathcal{U}$ is a pair $\langle \theta, \Psi' \rangle$ of a substitution and a collection of type assignments to term variables such that

1. θ is type preserving with respect to $\mathcal{N} \cup \Theta_0 \cup \Psi'$,
2. $\mathit{supp}(\theta) \cap \mathbb{N} = \emptyset$,
3. for any x if $x : \alpha \in \Psi$ and $x : \alpha' \in \mathit{ctx}(\theta) \uplus \Psi'$ then $\alpha = \alpha'$, and
4. for each i, $1 \leq i \leq n$, expressions E_i and E'_i from $E_i = E'_i$ are such that $E_i[\![\theta]\!] = E'_i[\![\theta]\!]$.

Note that the typing contraints validate the use of the notation $E_i[\![\theta]\!]$ and $E'_i[\![\theta]\!]$.

Given an atomic term R, we can determine its instances that have a particular head h through the unification of R with h applied to a sequence of fresh variables. We will use this idea to narrow down the set of instances of a sequent that must be considered once we have determined what the head of the term in an atomic goal of the form $\{G \vdash R : P\}$ must be. However, we must first build into our notion of a fresh variable the ability to instantiate it with nominal constants appearing in the sequent. We do this below by using the mechanism of raising.

Definition 4.17 (Generalized Variables)**.** Let Ψ be a finite set of arity typing assignments to term variables and let $\mathbb{N}$ be a finite subset of $\mathcal{N}$. Further, let $n_1, \ldots, n_k$ be a listing of the nominal constants in $\mathbb{N}$ and let $\alpha_1, \ldots, \alpha_k$ be the respective types of these constants. Then, for any variable z that does not appear in Ψ, $z : \alpha_1 \to \cdots \to \alpha_k \to \beta$ is said to be a variable of arity type β away from Ψ and raised over $\mathbb{N}$. Moreover, $(z\ n_1\ \ldots\ n_k)$ is said to be the generalized variable term corresponding to z.

The following lemma now formalizes the described refinement of the sequent.

Lemma 4.9. *Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega, F \longrightarrow F'$ be a well-formed sequent with F being the formula $\{G \vdash R : P\}$. Further, let θ be a term substitution that together with a context substitution σ identifies a closed instance of $\mathcal{S}$ and is such that, for the head h that is assigned the type $\Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P'$ by Σ or $G[\![\theta]\!][\sigma]$, $R[\![\theta]\!] = (h\ M_1\ \ldots\ M_n)$ and*

$$P[\![\theta]\!] = P'[\![\{\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_n, M_n, (A_n)^- \rangle\}]\!].$$

Finally, for each i, $1 \leq i \leq n$, let $z_i : \alpha'_i$ be a distinct variable of type $(A_i)^-$ away from Ψ and raised over $\mathbb{N}$, and let t_i be the generalized variable term corresponding to z_i. Then $\langle \theta, \emptyset \rangle$ is a solution to the unification problem

$$\begin{array}{l} \langle \mathbb{N}; \Psi \cup \{z_1 : \alpha'_1, \ldots, z_n : \alpha'_n\}; \\ \qquad\qquad \big\{P = P'[\![\{\langle x_1, t_1, (A_1)^- \rangle, \ldots, \langle x_n, t_n, (A_n)^- \rangle\}]\!], R = (h\ t_1\ \ldots\ t_n)\big\}\rangle. \end{array}$$

Proof. As θ and σ identify a closed instance of $\mathcal{S}$ is must be that $\langle \theta, \emptyset \rangle$ is substitution compatible with $\mathcal{S}$ and σ is appropriate for $\mathcal{S}[\![\Psi]\!]_\emptyset$. Therefore, the first three clauses of the definition for a solution will be satisfied by $\langle \theta, \emptyset \rangle$. The final clause of the definition is satisfied by the assumptions $P[\![\theta]\!] = P'[\![\{\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_n, M_n, (A_n)^- \rangle\}]\!]$ and $R[\![\theta]\!] = (h\ M_1\ \ldots\ M_n)$. Therefore $\langle \theta, \emptyset \rangle$ is a solution to the given unification problem. $\square$

The reduction of a sequent is based on lifting the observations of Theorem 2.12 to the analysis of atomic formulas. Since such an analysis must be driven by the structure of the LF type, reduction is only sensible when the atomic formula in question is one in which the term is atomic, and the head of the application is either a constant or a nominal constant which is bound in the explicit bindings of the context expression. It is with this type that we identify typing judgements for each argument term, and replace the original assumption formula with a set of formulas determined by this analysis.

Definition 4.18. [Reducing a Sequent] Let $F = \{G \vdash h\ M_1 \ldots M_n : P\}$ be a formula appearing in the well-formed sequent $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega, F \longrightarrow F'$, where h is assigned LF type $A = \Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P'$ in Σ or the explicit bindings in G relative to $\mathcal{S}$. Letting $A'_i = A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{n-1}, M_{n-1}, (A_{n-1})^-\rangle\}]\!]$ for each i, $1 \leq i \leq n$, the sequent obtained by decomposing the assumption formula F based on the type A is defined to be

$$\mathbb{N}; \Psi; \Xi; \Omega, \left\{G \vdash M_1 : A'_1\right\}, \ldots, \left\{G \vdash M_n : A'_n\right\} \longrightarrow F'.$$

This sequent is denoted by $\mathit{ReduceSeq}\,(\mathcal{S}, F)$. Note that the well-formedness of $\mathcal{S}$ justifies the use of the notation $A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{n-1}, M_{n-1}, (A_{n-1})^-\rangle\}]\!]$.

The following lemma expresses the soundness of the idea of reducing a sequent. Additionally, it identifies a measure with atomic formulas that diminishes with the replacements effected by a reduction step; this property will be useful in formulating an induction rule in Section 4.4.

Lemma 4.10. *Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega, F \longrightarrow F'$ be a well-formed sequent with F an atomic formula of the form $\{G \vdash R : P\}$. Further, let h be assigned type $\Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P'$ in either Σ or the explicit bindings in G relative to $\mathcal{S}$ and for each i, $1 \leq i \leq n$, let $z_i : \alpha'_i$ be a distinct variable of type $(A_i)^-$ away from Ψ and raised over $\mathbb{N}$, and let t_i be the generalized variable term corresponding to z_i. Finally, let $\mathcal{U}$ be the unification problem*

$$\langle \mathbb{N}; \Psi \cup \{z_1 : \alpha'_1, \ldots, z_n : \alpha'_n\}; \\ \left\{P = P'[\![\{\langle x_1, t_1, (A_1)^-\rangle, \ldots, \langle x_n, t_n, (A_n)^-\rangle\}]\!], R = (h\ t_1\ \ldots\ t_n)\right\}\rangle.$$

Then any solution to $\mathcal{U}$ is substitution compatible with $\mathcal{S}$. Further, for any $\langle \theta, \Psi_\theta\rangle$ that is a solution to $\mathcal{U}$ and θ_r that is a raising substitution associated with the application of θ to $\mathcal{S}$ relative to Ψ_θ, there must be terms $M_1, \ldots, M_n$ such that the following hold:

1. *$R[\![\theta]\!][\![\theta_r]\!]$ is $(h\ M_1\ \ldots\ M_n)$.*

2. *For any* θ' *and* σ' *identifying a closed instance of* $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}$, *if* $\{G \vdash R : P\}\ [\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma']$ *is valid and there is a derivation for* $(G \vdash_\Sigma R \Leftarrow P)[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma']$ *of height* k*, then for each* i*,* $1 \leq i \leq n$*, letting* $A'_i = A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!]$*, it must be the case that* $(\{G[\![\theta]\!][\![\theta_r]\!] \vdash M_i : A'_i\})[\![\theta']\!][\sigma']$ *is valid and that there is a derivation of height less than* k *for* $(G[\![\theta]\!][\![\theta_r]\!] \vdash_\Sigma M_i \Leftarrow A'_i)[\![\theta']\!][\sigma']$.

3. *There is an* $\mathcal{S}'$ *such that* $\mathcal{S}' = \mathit{ReduceSeq}\,(\mathcal{S}[\![\theta]\!]_{\Psi_\theta}, F[\![\theta]\!][\![\theta_r]\!])$ *and* $\mathcal{S}'$ *is valid only if* $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}$ *is.*

Proof. A straightforward examination of Definitions 4.3 and 4.16 suffices to verify that solutions to $\mathcal{U}$ must be substitution compatible with $\mathcal{S}$. Any solution $\langle \theta, \Psi_\theta\rangle$ to the unification problem $\mathcal{U}$ must be such that $R[\![\theta]\!] = (h\ t_1 \ldots t_n)[\![\theta]\!]$. From this it follows that $R[\![\theta]\!][\![\theta_r]\!] = (h\ t_1 \ldots t_n)[\![\theta]\!][\![\theta_r]\!]$. Since h is unaffected by substitutions, it is easy to see that $(h\ t_1 \ldots t_n)[\![\theta]\!][\![\theta_r]\!] = (h\ (t_1[\![\theta]\!][\![\theta_r]\!])\ \ldots\ (t_n[\![\theta]\!][\![\theta_r]\!]))$. Picking M_i to be the term $t_i[\![\theta]\!][\![\theta_r]\!]$ for each i, $1 \leq i \leq n$, we see that clause (1) in the lemma is satisfied.

For the second clause we note first that the typing judgements in question must be closed and hence the consideration is meaningful. Consider an arbitrary closed instance of $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}$ identified by θ' and σ. If $F[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma]$ is a valid formula then using the definition of validity as well as clause (1) in the lemma we can extract a derivation for $\vdash_\Sigma G[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma]$ `ctx` and a derivation of height k for $G[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma] \vdash_\Sigma (h\ M_1 \ldots M_n)[\![\theta']\!] \Leftarrow P[\![\theta]\!][\![\theta_r]\!][\![\theta']\!]$. Application of Theorems 2.12 and 2.3 are then sufficient to conclude that there is a derivation of height less than k for $(G[\![\theta]\!][\![\theta_r]\!] \vdash_\Sigma M_i \Leftarrow A'_i)[\![\theta']\!][\sigma]$. For us to be able to conclude that, for each i, $1 \leq i \leq n$, the formula $(\{G[\![\theta]\!][\![\theta_r]\!] \vdash M_i : A'_i\})[\![\theta']\!][\sigma]$ is valid, it only remains to show that there is a derivation for $G[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma] \vdash_\Sigma A'_i[\![\theta']\!]$ `type`. However, this has been done in the proof of Theorem 2.12.

We finish by proving the third clause. From clause (1) we know that $R[\![\theta]\!][\![\theta_r]\!]$ will be of the form $(h\ M_1 \ldots M_n)$, thus by the definition of *ReduceSeq* there must exist an $\mathcal{S}'$ such that $\mathcal{S}' = \mathit{ReduceSeq}\,(\mathcal{S}[\![\theta]\!]_{\Psi_\theta}, F[\![\theta]\!][\![\theta_r]\!])$. Suppose $\mathcal{S}'$ is valid. Consider an arbitrary closed instance of $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}$ identified by θ' and σ. These same θ' and σ also identify a closed instance of $\mathcal{S}'$ as these sequents only differ in that the assumption formula $F[\![\theta]\!][\![\theta_r]\!]$ has been replaced with the collection of reduced formulas $\{\{G[\![\theta]\!][\![\theta_r]\!] \vdash M_i : A_i'\} \mid 1 \leq i \leq n\}$. If any formula in the set of assumption formulas of $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}[\![\theta']\!]_\emptyset[\sigma]$ were not valid then this instance would be vacuously valid, so suppose all such formulas are valid. Then in particular, $F[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma]$ must be valid and thus by clause (2), for each i, $1 \leq i \leq n$, the formula $\{G[\![\theta]\!][\![\theta_r]\!] \vdash M_i : A_i'\}\,[\![\theta']\!][\sigma]$ will be valid. But then all of the assumption formulas of $\mathcal{S}'[\![\theta']\!][\sigma]$ must be valid and since this is a closed instance of a valid sequent we can conclude that the goal formula $F'[\![\theta]\!][\![\theta_r]\!][\![\theta']\!][\sigma]$ is valid. Therefore any closed instance of $\mathcal{S}[\![\theta]\!]_{\Psi_\theta}$ will be valid, and clause (3) in the lemma is satisfied. □

Lemmas 4.9 and 4.10 yield the following possibility for analyzing the derivability of a sequent with $\{G \vdash R : P\}$ as an atomic formula: we use the unification problem identified in Lemma 4.9 to limit the collection of closed term substitutions for the sequent and we analyze the derivability of the reduced sequent under these substitutions. Unfortunately, this approach would not be very effective in practice. What we would like to do instead is to use the unification problem directly to generate the collection of substitutions to be considered. Moreover, we would like to be able to limit the substitutions even from this set that actually need to be considered. Towards the latter end, we introduce the idea of a covering set of solutions to a unification problem. The next three definitions culminate in a formulation of this notion.

Definition 4.19 (Restricted Substitutions)**.** The restriction of a substitution θ to the arity typing context Ψ is the substitution $\{\langle x, M, \alpha\rangle \mid \langle x, M, \alpha\rangle \in \theta \text{ and } x : \alpha' \in \Psi\}$. We denote this substitution by $\theta|_\Psi$.

Definition 4.20 (Covering Substitutions)**.** Let Ψ, Ψ_1 and Ψ_2 be collections of type assignments to term variables, and let θ_1 and θ_2 be substitutions that are arity type preserving with respect to $\mathcal{N} \cup \Theta_0 \cup \Psi_1$ and $\mathcal{N} \cup \Theta_0 \cup \Psi_2$, respectively. Then $\langle \theta_2, \Psi_2\rangle$ is said to cover $\langle \theta_1, \Psi_1\rangle$ relative to Ψ if there exists a pair $\langle \theta_3, \Psi_3\rangle$ of a substitution and a collection of arity type assignments to term variables such that

1. θ_3 is type preserving with respect to $\mathcal{N} \cup \Theta_0 \cup \Psi_3$,

2. for any $x : \alpha \in \Psi_2$, if $x : \alpha' \in ctx(\theta_3) \uplus \Psi_3$ then $\alpha = \alpha'$, and

3. The substitutions $\theta_1|_\Psi$ and $(\theta_3 \circ \theta_2)|_\Psi$ are identical.

Note that the second condition ensures that $\mathcal{N} \cup \Theta_0 \cup ((\Psi_2 \setminus ctx(\theta_3)) \cup \Psi_3)$ determines a valid arity context and that θ_2 and θ_3 are arity type compatible with respect to this context. Thus, the composition of θ_2 and θ_3 in the third condition is well-defined.

Definition 4.21 (Covering Set of Solutions)**.** A collection S of solutions to a unification problem $\mathcal{U} = \langle \mathbb{N}; \Psi; \mathcal{E} \rangle$ is said to be *covering set of solutions for* $\mathcal{U}$ if every solution to $\mathcal{U}$ is covered by some solution in S relative to Ψ.

We now show the soundness of using the reduced forms of a sequent generated by just a covering set of solutions for the relevant unification problem in carrying out an analysis of the derivability of the sequent.

Lemma 4.11. *Suppose that θ_1 and σ identify a closed instance $\mathcal{S}'$ of a well-formed sequent $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F'$. Suppose further that $\langle \theta_2, \Psi_2 \rangle$ is substitution compatible with $\mathcal{S}$ and such that it covers $\langle \theta_1, \emptyset \rangle$ relative to Ψ. Then there is a term substitution θ that together with σ identifies a closed instance of $\mathcal{S}[\![\theta_2]\!]_{\Psi_2}$ that is valid if and only if $\mathcal{S}'$ is.*

Proof. We argue below that, under the assumptions of the lemma, there is a substitution θ_3 and a term variable context Ψ_3 such that $\langle \theta_3, \Psi_3 \rangle$ is substitution compatible with $\mathcal{S}[\![\theta_2]\!]_{\Psi_2}$ and for any term M such that $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M : \alpha$ is derivable it is the case that $M[\![\theta_2]\!][\![\theta_{2r}]\!][\![\theta_3]\!] = M[\![\theta_1]\!]$, where θ_{2r} is the raising substitution associated with the application of θ_2 to $\mathcal{S}$ relative to Ψ_2. It follows from this that the formulas and context types appearing in $S[\![\theta_2]\!]_{\Psi_2}[\![\theta_3]\!]_{\Psi_3}$ must be identical to the ones in $S[\![\theta_1]\!]_\emptyset$. It is then easily seen that θ_3 can be extended into a substitution θ that together with σ identifies a closed instance of $\mathcal{S}[\![\theta_2]\!]_{\Psi_2}$ whose formulas are identical to those of $\mathcal{S}'$. The lemma is an immediate consequence.

Since $\langle \theta_2, \Psi_2 \rangle$ covers $\langle \theta_1, \emptyset \rangle$ relative to Ψ there exists some $\langle \theta_4, \Psi_4 \rangle$ satisfying the conditions of Definition 4.20. We claim that we can further assume of this θ_4 that (1) $ctx(\theta_4) = (\Psi \setminus ctx(\theta_2)) \cup \Psi_2$ and (2) $supp(\theta_4) \cap \mathbb{N} = \emptyset$. The first of these may be violated because θ_4 may not instantiation some variables from Ψ_2 and it may also include instantiations for variables which are not contained in $(\Psi \setminus ctx(\theta_2)) \cup \Psi_2$. For the former we extend θ_4 with $\langle x, x, \alpha \rangle$ and the Ψ_4 with $x : \alpha$. For the latter we simply drop the instantiation from θ_4. It is straightforward to conclude that such changes to θ_4 will not violate any of the conditions of Definition 4.20. To address the second condition, consider a permutation π which renames nominal constants in $supp(\theta_4) \cap \mathbb{N}$ to new names chosen away from $\mathbb{N} \cup supp(\theta_1) \cup supp(\theta_2)$. The first two conditions of Definition 4.20 are obviously satisfied by $\pi.\theta_4$ and Ψ_4, so consider an arbitrary $\langle x, M, \alpha \rangle$ in $\theta_1|_\Psi$. By the definition of a composition of substitutions, since this same $\langle x, M, \alpha \rangle$ must appear in $(\theta_4 \circ \theta_2)|_\Psi$ either (1) $\langle x, M, \alpha \rangle \in \theta_4$ or (2) $M'[\![\theta_4]\!] = M$ for $\langle x, M', \alpha \rangle \in \theta_2$. Observe that since $\langle \theta_1, \emptyset \rangle$ and $\langle \theta_2, \Psi_2 \rangle$ are both substitution compatible with $\mathcal{S}$, neither substitution will contain any instances of nominal constants appearing in $\mathbb{N}$. Thus for the former case, $\pi.M = M$ and $\langle x, M, \alpha \rangle \in \pi.\theta_4$. For the latter case, a simple inductive argument on the structure of terms permits us to conclude that $\pi.(M'[\![\theta_4]\!]) = (\pi.M')[\![\pi.\theta_4]\!]$. But neither M nor M' contain any nominal constants from $\mathbb{N}$ and thus $M'[\![\pi.\theta_4]\!] = M$. Therefore $\theta_1|_\Psi$ is also identical to $(\pi.\theta_4 \circ \theta_2)|_\Psi$. Let θ_3' and Ψ_3 denote the $\pi.\theta_4$ and Ψ_4 which satisfy both (1) and (2).

We now use Theorem 4.4 to obtain a "raised" version of θ_3' that together with Ψ_3 will constitute the pair $\langle \theta_3, \Psi_3 \rangle$ that we desired at the outset. Specifically, the theorem allows us to conclude that there is a substitution θ_3 satisfying the following properties:

1. $supp(\theta_3)$ is disjoint from $\mathbb{N} \cup supp(\theta_2)$,

2. $ctx(\theta_3)$ is identical to the raised version of $(\Psi \setminus ctx(\theta_2)) \cup \Psi_2$ corresponding to the raising substitution θ_{2r},

3. θ_3 is arity type preserving with respect to $\mathcal{N} \cup \Theta \cup \Psi_3$, and

4. for every term M such that $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M : \alpha$, $M[\![\theta_2]\!][\![\theta_3']\!] = M[\![\theta_2]\!][\![\theta_{2r}]\!][\![\theta_3]\!]$.

The argument for the first three of these properties is obvious. For the last property, we observe, using Theorem 2.4, that $(\mathbb{N} \cup supp(\theta_2)) \cup \Theta_0 \cup ((\Psi \setminus ctx(\theta_2)) \cup \Psi_2) \vdash_{at} M[\![\theta_2]\!] : \alpha$ has a derivation under the condition described; Theorem 4.4 can then be invoked in an obvious way. It follows immediately from the first three properties that $\langle \theta_3, \Psi_3 \rangle$ is substitution compatible with $\mathcal{S}[\![\theta_2]\!]_{\Psi_2}$. It therefore only remains to show that for every M such that $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M : \alpha$, it is the case that $M[\![\theta_1]\!] = M[\![\theta_2]\!][\![\theta_{2r}]\!][\![\theta_3]\!]$. An easy inductive argument shows that for any M of the kind described and any θ, if $M[\![\theta]\!] = M'$ is derivable exactly when $M[\![\theta|_\Psi]\!] = M'$ is derivable. It follows from this that $M[\![\theta_1]\!] = M[\![\theta_2]\!][\![\theta_3']\!]$. Property (4) then yields the desired result. $\square$

We now use the observations in Lemmas 4.8, 4.9, 4.10 and 4.11 to describe a complete analysis of the derivability of a sequent around an atomic assumption formula.

Definition 4.22 (Cases Elaboration)**.** Let $\mathcal{S} = \mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F'$ be a well-formed sequent, let $F = \{G \vdash R : P\}$ be a formula in Ω and let $h : \Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P'$ be a type assignment in Σ or in the explicit bindings in G. Further, for each i, $1 \leq i \leq n$, let $z_i : \alpha_i$ be a distinct variable of type $(A_i)^-$ away from Ψ and raised over $\mathbb{N}$, and let t_i be the generalized variable term corresponding to z_i. Finally, let $\mathcal{U}$ be the unification problem

$$\langle \mathbb{N}; \Psi \cup \{z_1 : \alpha_1, \ldots, z_n : \alpha_n\}; \\ \{P = P'[\![\{\langle x_1, t_1, (A_1)^- \rangle, \ldots, \langle x_n, t_n, (A_n)^- \rangle\}]\!], R = (h\ t_1\ \ldots\ h_n)\}\rangle$$

and let C be a covering set of solutions for $\mathcal{U}$. Then the *analysis of $\mathcal{S}$ based on F and h* is denoted by $Cases\,(\mathcal{S}, F, h : A)$ and is given by the set of sequents

$$\left\{ ReduceSeq\left(\mathcal{S}[\![\theta]\!]_{\Psi_\theta}, F'\right) \;\middle|\; \begin{array}{l} \langle \theta, \Psi_\theta \rangle \in C \text{ and } F' \text{ is the formula in } \mathcal{S}[\![\theta]\!]_{\Psi_\theta} \\ \text{resulting from } F \end{array} \right\}.$$

If $\mathcal{S}$ is a well-formed sequent and $F = \{G \vdash R : P\}$ is an assumption formula in $\mathcal{S}$, then the *complete analysis of $\mathcal{S}$ based on F* is the set of sequents

$$\bigcup \{ Cases\,(\mathcal{S}', F', h : A) \mid (\mathcal{S}';\ h : A) \in Heads(\mathcal{S}, G) \\ \text{and } F' \text{ is the formula in } \mathcal{S}' \text{ resulting from } F\}.$$

This collection is denoted by $\mathit{AllCases}\,(\mathcal{S}, F)$. Note that the notations $\mathit{Cases}\,(\mathcal{S}, F, h : A)$ and $\mathit{AllCases}\,(\mathcal{S}, F)$ are both ambiguous—for instance, the first notation leaves out mention of the covering set of solutions that plays a role in generating the set it denotes. We will assume them to denote any of the set of sequents that can be generated in the respective ways described in this definition.

We show first that all the sequents in the set yielded by *AllCases* will be well-formed.

Theorem 4.18. *If $\mathcal{S}$ is a well-formed sequent and F is an atomic assumption formula appearing in $\mathcal{S}$, then every sequent in $\mathit{AllCases}\,(\mathcal{S}, F)$ is well-formed.*

Proof. Let $F = \{G \vdash R : P\}$. By Lemma 4.7 we know that every $\langle \mathcal{S}', h : A\rangle \in \mathit{Heads}(\mathcal{S}, G)$ is such that $\mathcal{S}'$ is well-formed and letting $\mathbb{N}'$ and Ψ' be the support set and arity typing context of $\mathcal{S}'$ respectively, $\mathbb{N}' \cup \Theta_0 \cup \Psi' \vdash_{ak} A\ \mathit{type}$ has a derivation. Letting F' be the formula from $\mathcal{S}'$ corresponding to F, we will denote the unification problem from $\mathit{Cases}\,(\mathcal{S}', F', h : A)$ by $\mathcal{U}_{(\mathcal{S}', h:A)}$. We can conclude by Theorem 4.5 that the application of any solution for $\mathcal{U}_{(\mathcal{S}', h:A)}$ to $\mathcal{S}'$ must be well-formed. We observe that for any formula $\{G \vdash h\ M_1 \ldots M_n : P\}$ which is well-formed with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ and Ξ^-, if $h : \Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P'$ appears in Σ or the explicit bindings of G then, for $1 \leq i \leq n$, we can extract derivations that $\left\{G \vdash M_i : A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!]\right\}$ are well-formed formulas with respect to the same $\mathbb{N} \cup \Theta_0 \cup \Psi$ and Ξ^-. From this observation we conclude that the reduced form of each sequent obtained by the application of a solution to $\mathcal{U}_{(\mathcal{S}', h:A)}$ will be well-formed, and therefore that all sequents in $\mathit{AllCases}\,(\mathcal{S}, F)$ must be well-formed. □

We show next that the validity of every sequent in the result of *AllCases* ensures the validity of the original sequent.

Theorem 4.19. *Let $\mathcal{S}$ be a well-formed sequent and let F be an atomic assumption formula in $\mathcal{S}$. If all the sequents in $\mathit{AllCases}\,(\mathcal{S}, F)$ are valid then $\mathcal{S}$ must be valid.*

Proof. Let $F = \{G \vdash R : P\}$. Consider an arbitrary closed instance of the sequent $\mathcal{S}$ identified by θ and σ. If any of the assumption formulas of $\mathcal{S}[\![\theta]\!]_{\emptyset}[\sigma]$ were not valid then this closed instance is vacuously valid, so it remains only to consider those instances for which they are all valid. If all these assumption formulas are valid, then $F[\![\theta]\!][\sigma]$ is a valid formula. Therefore by Lemma 4.8 there will exist a pair $\langle \mathcal{S}', h : A\rangle \in \mathit{Heads}(\mathcal{S}, G)$ such that, letting F' be the formula in $\mathcal{S}'$ corresponding to F, $h : A$ appears in Σ or among the explicit bindings of the context expression of F' with respect to $\mathcal{S}'$, and there is a closed instance of $\mathcal{S}'$ identified by θ' and σ' which is equivalent to $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ under some permutation π. Clearly then, if the closed instance $\mathcal{S}'[\![\theta']\!]_\emptyset[\sigma']$ is valid, $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ will be valid by Theorem 4.11. By Lemma 4.9 $\langle \theta', \emptyset\rangle$ must be a solution to the unification problem corresponding to the head pair $\langle \mathcal{S}', h : A\rangle$, and so there must be some solution $\langle \theta_1, \Psi_1\rangle$ in the covering set of solutions C used by *Cases* which covers it. By Lemma 4.10 then, there is some sequent $\mathcal{S}''$ such that, letting F'' be the formula in $\mathcal{S}'[\![\theta_1]\!]_{\Psi_1}$ corresponding to F', $\mathcal{S}'' = \mathit{ReduceSeq}\,(F'', \mathcal{S}'[\![\theta_1]\!]_{\Psi_1})$ and $\mathcal{S}'[\![\theta_1]\!]_{\Psi_1}$ is valid if $\mathcal{S}''$ is. By assumption every sequent in $\mathit{AllCases}\,(\mathcal{S}, F)$ is valid, and so $\mathcal{S}''$ will be valid and thus $\mathcal{S}'[\![\theta_1]\!]_{\Psi_1}$ as well. But by Lemma 4.11, there exists some $\langle \theta_2, \emptyset\rangle$ which together with σ' identifies a closed instance of $\mathcal{S}'[\![\theta_1]\!]_{\Psi_1}$ which is valid if and only if $\mathcal{S}'[\![\theta']\!]_\emptyset[\sigma']$ is. Thus by the validity of $\mathcal{S}'[\![\theta_1]\!]_{\Psi_1}$ the closed instance $\mathcal{S}'[\![\theta']\!]_\emptyset[\sigma']$ will be valid, and we have already seen that this will ensure that $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ is valid. Therefore $\mathcal{S}$ will be valid, since all of its closed instances are valid. □

4.3.2 Proof Rules that Introduce Atomic Formulas

We are finally in a position to describe rules in our logic that internalize the analysis of typing derivations in LF. We do this in Figure 4.5. Reasoning about atomic formulas over atomic terms, we base our reasoning step on the result of Theorem 2.12. When analysing a goal formula we require that the head of the atomic term is known and thus a single structure is possible; when an assumption formula we rely on the analysis of *AllCases* to identify all the ways such judgements might have been derived. When reasoning about an atomic formula for an abstraction term, we introduce a new nominal constant to represent the fresh binding of the LF typing rule for abstractions and extend the context of the formula with this new binding. In LF this name is distinct and we capture this requirement

$$\frac{\mathit{AllCases}\,(\mathbb{N};\Psi;\Xi;\Omega,\{G\vdash R:P\}\longrightarrow F,\{G\vdash R:P\})}{\mathbb{N};\Psi;\Xi;\Omega,\{G\vdash R:P\}\longrightarrow F}\ \mathit{atm\text{-}app\text{-}L}$$

$$\frac{\begin{array}{c} h:\Pi x_1{:}A_1.\ \ldots\Pi x_n{:}A_n.\,P\in\Sigma \text{ or the explicit bindings in } G \\ \{G\vdash N:B\}\in\Omega \qquad P[\![\{\langle x_1,M_1,(A_1)^-\rangle,\ldots,\langle x_n,M_n,(A_n)^-\rangle\}]\!]=P' \\ \left\{\begin{array}{l} \mathbb{N};\Psi;\Xi;\Omega\longrightarrow \\ \quad \big\{G\vdash M_i:A_i[\![\{\langle x_1,M_1,(A_1)^-\rangle,\ldots,\langle x_{i-1},M_{i-1},(A_{i-1})^-\rangle\}]\!]\big\} \\ \hfill \mid\ 1\le i\le n \end{array}\right\} \end{array}}{\mathbb{N};\Psi;\Xi;\Omega\longrightarrow\{G\vdash h\ M_1\ldots M_n:P'\}}\ \mathit{atm\text{-}app\text{-}R}$$

$$\frac{\begin{array}{c} n\notin \mathit{dom}\,(\mathbb{N}) \\ \Xi'=\left\{\begin{array}{ll} (\Xi\setminus\{\Gamma{\uparrow}\mathbb{N}_\Gamma:\mathcal{C}[\mathcal{G}]\})\cup\big\{\Gamma{\uparrow}(\mathbb{N}_\Gamma,n:(A_1)^-):\mathcal{C}[\mathcal{G}]\big\} & \text{if } \Gamma \text{ in } G \\ \Xi & \text{otherwise} \end{array}\right. \\ \mathbb{N},n:(A_1)^-;\Psi;\Xi'; \\ \quad \Omega,\big\{G,n:A_1\vdash M[\![\{\langle x,n,(A_1)^-\rangle\}]\!]:A_2[\![\{\langle x,n,(A_1)^-\rangle\}]\!]\big\}\longrightarrow F \end{array}}{\mathbb{N};\Psi;\Xi;\Omega,\{G\vdash \lambda x.\,M:\Pi x{:}A_1.\,A_2\}\longrightarrow F}\ \mathit{atm\text{-}abs\text{-}L}$$

$$\frac{\begin{array}{c} n\notin \mathit{dom}\,(\mathbb{N}) \\ \Xi'=\left\{\begin{array}{ll} (\Xi\setminus\{\Gamma{\uparrow}\mathbb{N}_\Gamma:\mathcal{C}[\mathcal{G}]\})\cup\big\{\Gamma{\uparrow}(\mathbb{N}_\Gamma,n:(A_1)^-):\mathcal{C}[\mathcal{G}]\big\} & \text{if } \Gamma \text{ in } G \\ \Xi & \text{otherwise} \end{array}\right. \\ \mathbb{N},n:(A_1)^-;\Psi;\Xi';\Omega\longrightarrow \\ \quad \big\{G,n:A_1\vdash M[\![\{\langle x,n,(A_1)^-\rangle\}]\!]:A_2[\![\{\langle x,n,(A_1)^-\rangle\}]\!]\big\} \end{array}}{\mathbb{N};\Psi;\Xi;\Omega\longrightarrow\{G\vdash \lambda x.\,M:\Pi x{:}A_1.\,A_2\}}\ \mathit{atm\text{-}abs\text{-}R}$$

Figure 4.5: Proof Rules Interpreting Atomic Formulas

in the logic by ensuring that no instantiations for existing eigenvariables or for context variables in the formula may use this name directly.

The following theorem shows that these rules require the proof of only well-formed sequents in constructing a proof of a well-formed sequent.

Theorem 4.20. *The following property holds for each rule in Figure 4.5: if the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then all the sequent premises must be well-formed.*

Proof. Consider each of the rules in Figure 4.5.

Case: *atm-app-L*

For this rule, the well-formedness of the premise sequents is assured by Theorem 4.18 given that the conclusion sequent is well-formed by assumption.

Case: *atm-app-R*

Let $\mathcal{S}_1$ …, $\mathcal{S}_n$ denote the premise sequents for A_1, …, A_n respectively, and assume that the conclusion sequent is well-formed. Since the context variable context and assumption formulas remain the same in these premise sequents, we can infer they are well-formed from the well-formedness of the conclusion sequent; what remains to be shown is that the goal formulas of these premise sequents are well-defined and will be well-formed. For each i, $1 \leq i \leq n$, let θ_i denote $\{\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^- \rangle\}$. By the well-formedness of the conclusion sequent, $\{G \vdash h\ M_1 \ldots M_n : P'\}$ is a well-formed formula with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ and Ξ^-. Thus there must exist a derivation of the judgement $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M_i : (A_i)^-$ for each i, $1 \leq i \leq n$. Therefore the substitutions θ_i will all be arity type preserving with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ and so the premise sequents will be well-defined. For each i, $1 \leq i \leq n$, let A'_i denote the type $A_i[\![\theta_i]\!]$. Since h is bound in Σ or the explicit bindings in G relative to the conclusion sequent, we further extract derivations of $\mathbb{N} \cup \Theta_0 \cup \Psi_i \vdash_{ak} A_i\ \mathit{type}$ for each i. By Theorem 2.6 $(A_i)^- = (A'_i)^-$, and therefore $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M_i : (A'_i)^-$ has a derivation for each i. A straightforward inner induction on i showing θ_i is arity type preserving with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ permits us to conclude through Theorem 2.4 that for each i, $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A'_i$ *type* is derivable. Thus we can construct a derivation for $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \{G \vdash M_i : A'_i\}$ *fmla* for each i, $1 \leq i \leq n$, and therefore each premise sequent $\mathcal{S}_i$ will clearly be well-formed.

Case: *atm-abs-L* **and** *atm-abs-R*

We first observe that the substitution $\{\langle x, n, (A_1)^- \rangle\}$ is obviously arity type preserving by construction and so the premise sequents will be well-defined. Let A'_2 denote the type $A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!]$ and M' denote the term $M[\![\{\langle x, n, (A_1)^- \rangle\}]\!]$. By the well-formedness of the conclusion sequent we know that

1. for each $\Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$, $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty has a derivation and
2. for each $F' \in \Omega \cup \{\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}, F\}$ (resp. $\Omega \cup \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}$ for *atm-abs-R*), $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash F'$ *fmla* has a derivation.

We first consider the well-formedness of the context variable types. For every binding $\Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ which does not appear in G, it is obvious by Theorem 4.2 that this context variable type will remain well-formed under the extended support set $\mathbb{N}, n : (A_1)^-$. For any Γ_i which appears in G, we observe that $\mathbb{N} \setminus \mathbb{N}_i$ is the same set as $(\mathbb{N}, n : (A_1)^-) \setminus (\mathbb{N}_i, n : (A_1)^-)$ and thus the declaration $\Gamma_i{\uparrow}(\mathbb{N}_i, n : (A_1)^-) : \mathcal{C}_i[\mathcal{G}_i]$ must also be well-formed.

We now consider the well-formedness of formulas in the sequent. We first observe that Ξ^- is the same as Ξ'^-, thus these sets can be used interchangeable. By Theorem 3.2 we easily determine that the formulas in $\Omega \cup \{F\}$ (resp. Ω for *atm-abs-R*) will all be well-formed relative to the extended support set $\mathbb{N}, n : (A_1)^-$ and Ξ'^-. It remains only to show the well-formedness of the formula $\{G, n : A_1 \vdash M' : A'_2\}$. By assumption $\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}$ is well-formed with respect to $\mathbb{N} \cup \Theta_0 \cup \Psi$ and Ξ^-, and therefore that G is a well-formed context, $\Pi x{:}A_1.\, A_2$ a well-formed type, and $\lambda x.\, M$ a well-formed term of type $(\Pi x{:}A_1.\, A_2)^-$ with respect to this same $\mathbb{N} \cup \Theta_0 \cup \Psi$ and Ξ^-. For a new $n : (A_1)^-$ then, we can extract derivations showing $(G, n : A_1)$ is a well-formed context, A'_2 a well-formed type, and M' a well-formed term of type $(A_2)^-$ with respect to $(\mathbb{N}, n : (A_1)^-) \cup \Theta_0 \cup \Psi$ and Ξ^-. But then obviously $\{G, n : A_1 \vdash M' : A'_2\}$ is well-formed with respect to $(\mathbb{N}, n : (A_1)^-) \cup \Theta_0 \cup \Psi$ and

Ξ'^{-}. Therefore the premise sequent must be well-formed. □

We now establish the soundness of these rules.

Theorem 4.21. *The following property holds for every instance of each of the rules in Figure 4.5: if the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and the premise sequents are valid, then the conclusion sequent must also be valid.*

Proof. Consider each rule in Figure 4.5.

Case: *atm-app-L*

The soundness of this rule is an immediate consequence of Theorem 4.19.

Case: *atm-app-R*

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ denote the premise sequents corresponding to $A_1, \ldots, A_n$ respectively and for each i, $1 \leq i \leq n$ let A'_i denote $A_i[\![\{\langle x_1, M_1[\![\theta]\!], (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}[\![\theta]\!], (A_{i-1})^-\rangle\}]\!]$. By assumption all of the premise sequents are valid. Consider an arbitrary closed instance of the conclusion sequent identified by θ and σ. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid then this instance would be vacuously valid, so assume they are all valid. This θ and σ clearly also identify closed instances for each $\mathcal{S}_i$. Since all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid, $\{G \vdash N : B\}[\![\theta]\!][\sigma]$ in particular must be valid and therefore there will exist a derivation for $\vdash_\Sigma G[\![\theta]\!][\sigma]\ \texttt{ctx}$. Also using the validity of $\Omega[\![\theta]\!][\sigma]$, the validity of the premise sequents ensures that for each i, $1 \leq i \leq n$, $\{G \vdash M_i : A'_i\}[\![\theta]\!][\sigma]$ is valid. Using Theorem 2.3 to permute the substitution application to A_i this formula can be written as $\left\{G[\![\theta]\!][\sigma] \vdash M_i[\![\theta]\!] : A_i[\![\theta]\!][\![\{\langle x_1, M_1[\![\theta]\!], (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}[\![\theta]\!], (A_{i-1})^-\rangle\}]\!]\right\}$. By assumption, h is bound in either Σ or the explicit bindings of G with respect to the conclusion sequent. Thus since $G[\![\theta]\!][\sigma]$ must be a well-formed LF context, there will exist a derivation in LF for $G[\![\theta]\!][\sigma] \vdash_\Sigma (\Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n. P)[\![\theta]\!]\ \texttt{type}$. Given the validity of the formulas expressed above, the substitution $\{\langle x_1, M_1[\![\theta]\!], (A_1)^-\rangle, \ldots, \langle x_n, M_{i-1}[\![\theta]\!], (A_n)^-\rangle\}$ is clearly arity type preserving with respect to $\mathcal{N} \cup \Theta_0$ and therefore by Theorem 2.4 and another use of Theorem 2.3 there is a derivation of $G[\![\theta]\!][\sigma] \vdash_\Sigma P'[\![\theta]\!]\ \texttt{type}$. We conclude by Theorem 2.12 that there is a derivation for $G[\![\theta]\!][\sigma] \vdash_\Sigma (h\ M_1 \ldots M_n)[\![\theta]\!] \Leftarrow P'[\![\theta]\!]$, in LF and it is then easy to see that $\{G \vdash h\ M_1 \ldots M_n : P'\}\,[\![\theta]\!][\sigma]$ will be valid. Thus the conclusion sequent must be valid.

Case: *atm-abs-L*

Let $\mathcal{S}$ denote the conclusion sequent and $\mathcal{S}'$ the premise sequent. Since $\mathcal{S}$ is well-formed and $n : (A_1)^- \notin \mathbb{N}$ we can conclude that n does not appear anywhere in $\mathcal{S}$. Suppose that $\mathcal{S}'$ is valid. Consider an arbitrary closed instance of $\mathcal{S}$ identified by θ and σ. It suffices to consider only those θ and σ which do not contain any uses of the nominal constant $n : (A_1)^-$ as any substitutions using this name can be obtained by applying a permutation π to some other substitution which does not, and by Theorem 4.11 the validity of the closed instance identified by the substitutions without n will then ensure the validity of the instance identified by the substitutions using n. Since n does not appear in either θ or σ, these substitutions must also identify a closed instance of the sequent $\mathcal{S}'$. If any formula in $(\Omega, \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\})[\![\theta]\!][\sigma]$ were not valid then this instance would be vacuously valid, so assume they are all valid. In particular then, the formula $\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}\,[\![\theta]\!][\sigma]$ is valid. By the definition of validity for formulas there are derivations in LF for $\vdash_\Sigma G[\![\theta]\!][\sigma]\ \mathtt{ctx}$, $G[\![\theta]\!][\sigma] \vdash_\Sigma (\Pi x{:}A_1.\, A_2)[\![\theta]\!][\sigma]\ \mathtt{type}$, and $G[\![\theta]\!][\sigma] \vdash_\Sigma (\lambda x.\, M)[\![\theta]\!][\sigma] \Leftarrow (\Pi x{:}A_1.\, A_2)[\![\theta]\!][\sigma]$. Since the nominal constant n cannot appear anywhere in these judgements we can extract from these derivations that the judgements $\vdash_\Sigma (G, n : A_1)[\![\theta]\!][\sigma]\ \mathtt{ctx}$, $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma]\ \mathtt{type}$, and $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma] \Leftarrow A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma]$ have derivations as well. Thus $\left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^-\rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\right\}\,[\![\theta]\!][\sigma]$ will be a valid formula. But then all of the assumption formulas of $\mathcal{S}'[\![\theta]\!]_\emptyset[\sigma]$ are valid and by the validity of this sequent we can conclude that $F[\![\theta]\!][\sigma]$ must be valid. Therefore the closed instance $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ is valid, and thus we can conclude that $\mathcal{S}$ is a valid sequent.

Case: *atm-abs-R*

Let $\mathcal{S}$ denote the conclusion sequent and $\mathcal{S}'$ the premise sequent. Since $\mathcal{S}$ is well-formed and $n : (A_1)^- \notin \mathbb{N}$ we can conclude that n does not appear anywhere in $\mathcal{S}$. Suppose that

$\mathcal{S}'$ is valid. Consider an arbitrary closed instance of $\mathcal{S}$ identified by θ and σ. As with *atm-abs-L* it will suffice to consider only those closed instances which do not use $n : (A_1)^-$ as permutations of valid closed instances must also be valid. This θ and σ will therefore also identify a closed instance of the premise sequent $\mathcal{S}'$. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid then this instance of $\mathcal{S}$ would be vacuously valid, so assume they are all valid. But then by the validity of $\mathcal{S}'$ the formula $\left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^-\rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\right\}[\![\theta]\!][\sigma]$ will be valid. Thus there must be LF derivations for

1. $\vdash_\Sigma (G, n : A_1)[\![\theta]\!][\sigma]$ $\mathtt{ctx}$,
2. $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma]$ $\mathtt{type}$, and
3. $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma] \Leftarrow A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!][\sigma]$.

From these we are able to construct LF derivations for the judgements $\vdash_\Sigma G[\![\theta]\!][\sigma]$ $\mathtt{ctx}$, $G[\![\theta]\!][\sigma] \vdash_\Sigma (\Pi x{:}A_1.\, A_2)[\![\theta]\!][\sigma]$ $\mathtt{type}$, and $G[\![\theta]\!][\sigma] \vdash_\Sigma (\lambda x.\, M)[\![\theta]\!][\sigma] \Leftarrow (\Pi x{:}A_1.\, A_2)[\![\theta]\!][\sigma]$. But then $\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}[\![\theta]\!][\sigma]$ must be valid by definition. Therefore the closed instance $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ is valid, and we can conclude $\mathcal{S}$ is a valid sequent. $\square$

4.4 An Annotation Based Scheme for Induction

In this section we build into the proof system a means for reasoning by induction on the height of LF derivations. The idea we use is borrowed from the Abella proof system, specialized to the context where atomic formulas encapsulate derivability in LF. In particular, we describe an annotation scheme that allows us to encode when an atomic formula represents an LF derivation that has a height less than that of the LF derivation represented by an atomic formula that appears in a formula being proved and, hence, when a property in which this atomic formula appears negatively can be assumed to hold in an inductive argument. In the first subsection we will introduce the extension of formula syntax to include annotations and define a new notion of semantics in relation to this syntax which is equivalent to the other semantics we have seen when no annotations occur. The second subsection will introduce an induction rule which uses annotations to capture strong induction on the height of LF derivations. To work with annotated formulas in reasoning we introduce alternative forms for atomic proof rules as well as *id* which are applicable when the formulas have annotations.

4.4.1 Extending Formula Syntax with Annotations

As mentioned above, we annotate particular atomic formulas to indicate relative heights associated with them. The annotations that we use go in pairs: @ and $*$, @@ and $**$, and so on. For ease we use $@^n$ (resp. $*^n$) to denote a sequence in which the character @ (resp. $*$) is repeated n times. We use $F^{@^i}$ on an atomic formula F to indicate that it has a certain height and F^{*^i} to indicate that it has a strictly smaller height; we will explain what a height means shortly. This height annotation is decreased whenever we decompose a derivation into sub-derivations based on its structure, as is done in the *atm-app-L* rule of the previous section.

To understand the meaning of the annotations recall first that an atomic formula $\{G \vdash M : A\}$, is valid if then there are LF derivations for $\vdash_\Sigma G$ $\mathtt{ctx}$, $G \vdash_\Sigma A$ $\mathtt{type}$ and $G \vdash_\Sigma M \Leftarrow A$. Each of these derivations in LF will have a particular height, and it is the height of the typing judgement $G \vdash_\Sigma M \Leftarrow A$ which forms the basis for our induction. Thus, when we talk of an atomic formula being restricted to a particular height or heights, we mean the height of the derivation of this typing judgement. In particular, the valid closed instances of the annotated atomic formula $\{G \vdash M : A\}^{@^i}$ are the ones for which the corresponding instances of $G \vdash_\Sigma M \Leftarrow A$ have derivations of height up to some particular size m, while the closed instances of the relatedly annotated formula $\{G' \vdash M' : A'\}^{*^i}$ will be valid only if the corresponding instances of $G' \vdash_\Sigma M' \Leftarrow A'$ have derivations of a height strictly smaller than m. Having available a denumerable collection of pairs of such annotations allows us to simultaneously relate the heights of different pairs of atomic formulas in this manner. We may of course also want to consider atomic formulas without any annotations. We use the notation $\{G \vdash M : A\}^{Ann}$ to denote a formula which may be unannotated

($\{G \vdash M : A\}$) or have an annotation ($\{G \vdash M : A\}^{@^n}$ or $\{G \vdash M : A\}^{*^n}$). Note that only the syntax of atomic formulas is extended with these annotations. In the remainder of this subsection we formalize the concepts of well-formedness and validity in the context of the formula syntax extended with annotations, and show that the proof system satisfies the same well-formedness and soundness properties with this extension.

The well-formedness of formulas containing annotations is determined essentially by looking at the formula after erasing the annotations.

Definition 4.23 (Well-formed Formulas with Annotations)**.** A formula F containing annotations is well-formed with respect to Θ and Ξ if the formula F' obtained by erasing all annotations in F is such that $\Theta; \Xi \vdash F'$ *fmla* holds.

Similar to the formulas, sequents containing annotations are well-formed if, ignoring the annotations, the sequent is well-formed as defined in Definition 4.2.

Definition 4.24 (Well-formed Sequents with Annotations)**.** A sequent $\mathcal{S}$ containing annotations is well-formed if the sequent $\mathcal{S}'$ obtained from $\mathcal{S}$ by erasing all annotations is well-formed.

We now define precisely the meaning of sequents which contain these annotations. A key part of this definition is describing an association between annotations and actual heights.

Definition 4.25 (Height Assignments)**.** A height assignment Υ will map each annotation $@^i$ to a particular height m_i, with the height restriction associated with $*^i$ inferred from the mapping for $@^i$. For a height assignment Υ the height assignment which is the same as Υ everywhere except that $@^i$ is mapped to m is represented by $\Upsilon[@^i \hookleftarrow m]$.

Annotated formulas are then interpreted relative to height assignments.

Definition 4.26 (Validity of Annotated Formulas)**.** We define validity only for closed annotated formulas that are well-formed, i.e, for formulas F such that $\mathcal{N} \cup \Theta_0; \emptyset \vdash F'$ *fmla* is derivable for the F' that is obtained by erasing the annotations in F. For such formulas, we first define formula validity with respect to a height assignment Υ, written $\Upsilon \models F$ `valid`, as follows:

- $\Upsilon \models \{G \vdash M : A\}^{Ann}$ `valid` holds if $\vdash_\Sigma G$ `ctx` and $G \vdash_\Sigma A$ `type` are derivable, and
  - $G \vdash_\Sigma M \Leftarrow A$ has a derivation if Ann is the empty annotation,
  - $G \vdash_\Sigma M \Leftarrow A$ has a derivation of height less than or equal to $\Upsilon(@^i)$ if Ann is $@^i$, and
  - $G \vdash_\Sigma M \Leftarrow A$ has a derivation of height less than $\Upsilon(@^i)$ if Ann is $*^i$.
- $\Upsilon \models \top$ `valid` holds.
- $\Upsilon \models \bot$ `valid` does not hold.
- $\Upsilon \models F_1 \supset F_2$ `valid` holds if $\Upsilon \models F_2$ `valid` holds in the case that $\Upsilon \models F_1$ `valid` holds.
- $\Upsilon \models F_1 \wedge F_2$ `valid` holds if both $\Upsilon \models F_1$ `valid` and $\Upsilon \models F_2$ `valid` hold.
- $\Upsilon \models F_1 \vee F_2$ `valid` holds if either $\Upsilon \models F_1$ `valid` or $\Upsilon \models F_2$ `valid` holds.
- $\Upsilon \models \Pi\,\Gamma : \mathcal{C}.F$ `valid` holds if $\Upsilon \models F[G/\Gamma]$ `valid` holds for every context expression G such that $\mathcal{N}; \emptyset \vdash \mathcal{C} \rightsquigarrow_{cs} G$ is derivable.
- $\Upsilon \models \forall x : \alpha.F$ `valid` holds if $\Upsilon \models F[\![\{\langle x, M, \alpha\rangle\}]\!]$ `valid` holds for every M such that $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ is derivable.
- $\Upsilon \models \exists x : \alpha.F$ `valid` holds if $\Upsilon \models F[\![\{\langle x, M, \alpha\rangle\}]\!]$ `valid` holds for some M such that $\mathcal{N} \cup \Theta_0 \vdash_{at} M : \alpha$ is derivable.

As in the case with Definition 3.2, the coherence of this definition is assured by Theorem 3.4.

Finally, the validity of a sequent containing annotations corresponds to the validity of each of its closed instances relative to every height assignment. In formalizing this idea, we assume the adaptation of the notions of the compatibility of term substitutions (Definition 4.3), the appropriateness of context substitutions (Definition 4.6) and the applications of these substitutions (Definitions 4.5 and 4.7) to annotated sequents that is obtained by ignoring the annotations on formulas. Further, we refer to every instance of an annotated well-formed sequent that is determined by term and context substitutions as in Definition 4.8 as one of its closed instances; the wellformedness of these closed instances follows easily from the results of Section 4.1.

Definition 4.27 (Validity of Annotated Sequents)**.** A well-formed sequent of the form $\mathbb{N}; \emptyset; \emptyset; \Omega \longrightarrow F$ is valid with respect to a height assignment Υ if $\Upsilon \models F$ `valid` holds whenever $\Upsilon \models F'$ `valid` holds for every $F' \in \Omega$. A well-formed sequent $\mathcal{S}$ is valid with respect to Υ if every closed instance of $\mathcal{S}$ is valid with respect to Υ. A well-formed sequent $\mathcal{S}$ is considered valid under the extended semantics if $\mathcal{S}$ is valid with respect to every height assignment.

While height assignments associate heights with every annotation, the assignments to only a finite subset of annotations matter in determining the validity of a formula or sequent. This observation is an immediate consequence of the following lemma.

Lemma 4.12. *If F is a formula in which the annotations $@^i$ and $*^i$ do not occur, then $\Upsilon \models F$ `valid` holds for a height assignment Υ if and only if $\Upsilon[@^i \hookleftarrow m] \models F$ `valid` holds for every choice of m. Similarly, if $\mathcal{S}$ is a sequent which the annotations $@^i$ and $*^i$ do not occur, then $\mathcal{S}$ is valid with respect to Υ if and only if $\mathcal{S}$ is valid with respect to $\Upsilon[@^i \hookleftarrow m]$ for every choice of m.*

Proof. The first clause is shown by a straightforward induction on the formation of F where in the base case we know that no atomic formula is annotated by $@^i$ or $*^i$ and therefore the m_i assigned to that annotation cannot play a role in determining its validity. The second clause follows from the first using the definition of validity for sequents of Definition 4.27. □

Our ultimate interest is in determining that formulas devoid of annotations are valid in the sense articulated in Definition 3.2. This also means that we are eventually interested in the validity of sequents devoid of annotations in the sense described in Definition 4.8.

However, in building in capabilities for inductive reasoning, we will consider rules that will introduce annotated formulas into sequents. Establishing the soundness of the resulting proof system requires us to refine the definition of validity for sequents to the one presented in Definition 4.27. That this is acceptable from the perspective of our eventual goal is the content of the following theorem.

Theorem 4.22. *A well-formed sequent that is devoid of annotations is valid in the sense of Definition 4.27 if and only if it is valid in the sense of Definition 4.8.*

Proof. We claim that if F is a closed, well-formed formula that contains no annotations, then F is valid in the sense of Definition 3.2 if and only if $\Upsilon \vDash F\ \texttt{valid}$ holds for any height assignment Υ. This claim is proved easily by induction on the formation of F: the base case is obvious given the definition of $\Upsilon \vDash \{G \vdash M : A\}\ \texttt{valid}$ and the other cases have a simple inductive structure. It can now be easily argued that a closed instance of a sequent devoid of annotations is valid in the sense of Definition 4.8 if and only if it is valid by virtue of Definition 4.26 with respect to any height assignment. The theorem is an obvious consequence of this. $\square$

Our attention henceforth will be on sequents that potentially contain annotated formulas. For this reason, absent qualifications, by "wellformedness" for formulas and sequents we shall mean by virtue of Definitions 4.23 and 4.24. Similarly, "validity" shall be interpreted by virtue of Definitions 4.26 and 4.27. Now, the proof rules that we have discussed up to this point that apply to non-atomic formulas and that are different from the *id* rule lift in an obvious way to the situation where formulas carry annotations. The *id* rule still applies as before when the formula that is the focus of the rule is unannotated. Similarly, no change is needed to the rules for atomic formulas when these formulas are unannotated. We show below that when these rules are interpreted in this manner, they continue to preserve wellformedness of sequents and to be sound, albeit with respect to the extended semantics. The extension of the *id* rule and the rules for atomic formulas to the situation when the focus formula is annotated needs some care. We take up this matter after the presentation of the induction rule.

Theorem 4.23. *The following properties hold for the lifted forms of the proof rules in Figure 4.2, Figure 4.4, the* cut *proof rule from Figure 4.3, and the* id *rule and the rules from Figure 4.5 when the formula they pertain to are unannotated:*

1. *If the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then the premise sequents must be well-formed.*

2. *If the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and all the premise sequents are valid, then the conclusion sequent must also be valid.*

Proof. The definition of wellformedness for annotated sequents is based on the original notion via the erasure of annotations. Hence, the first claim follows immediately from the earlier results for unannotated sequents. The second claim can be proved relative to an arbitrary height assignment; this then generalizes to all possible height assignments. For the proof rules not relating to atomic formulas, the heights which may be assigned to annotations can play no role in the soundness argument as there are no atomic formulas being interpreted within the proof. For the atomic proof rules the claim is restricted to the case where the formulas being analysed in the rules are not annotated, and thus again the height assignment has no impact on the previously presented argument for soundness. □

4.4.2 The Induction Proof Rule

The induction rule is presented in Figure 4.6. In this rule $\mathcal{Q}_i$ represents a sequence of context quantifiers or universal term quantifiers. There is also a proviso on the rule: the annotations $@^i$ and $*^i$ must be fresh, i.e., they must not already appear in the sequent that is the conclusion of the rule. Induction in this form is based on the induction principle for natural numbers applied to the heights of atomic formulas, which themselves encode the heights of LF derivations. We can view the premise of this rule as providing a proof schema

$$\frac{\mathbb{N};\Psi;\Xi;\Omega, \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{*^i} \supset \ldots \supset F_n))) \longrightarrow \atop \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))}\ ind$$

Figure 4.6: The Induction Proof Rule

for constructing an argument of validity for any m, and so by an inductive argument we conclude that it must be valid for all heights.

Given the definition of well-formedness for sequents containing annotations, the well-formedness of the conclusion sequent will clearly be sufficient to ensure that the premise sequent is well-formed.

Theorem 4.24. *If the conclusion sequent of an instance of the* ind *rule is well-formed, then the premise sequent must be well-formed.*

Proof. Clearly the well-formedness of each declaration in Ξ and each formula in Ω will hold given that these are the same in the premise sequent as in the conclusion sequent. Since both the formula $\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$ and the formula $\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{*^i} \supset \ldots \supset F_n)))$ are the same as the formula $\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))$ under the erasure of annotations, they are also obviously well-formed by the well-formedness of the conclusion sequent. □

A derivation of the premise sequent of the induction rule provides a schema for constructing a concrete, valid derivation for any closed instance of the conclusion sequent based on the height m of the assumption derivation in LF. Since this proof will be general with respect to the choice of m it ensures that the property holds for all natural numbers, and so the sequent without an annotated atomic formula will be valid. The soundness of this rule is shown by formalizing these ideas as a meta-level argument using induction on natural numbers. The following two lemmas are useful towards this end.

Lemma 4.13. *Let $@^i$ and $*^i$ be annotations which do not appear in any of the formulas $F_1, \ldots, F_{k-1}, F_{k+1}, \ldots F_n$, Υ a height assignment, and let m be a natural number. If*

$$\Upsilon[@ \hookleftarrow l] \vDash \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n))) \ \texttt{valid}$$

holds for every $l < m$, then

$$\Upsilon[@^i \hookleftarrow m] \vDash \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{*^i} \supset \ldots \supset F_n))) \ \texttt{valid}.$$

also holds.

Proof. Let $F = \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$. We will prove this result by induction on the formation of F measuring size by the number of quantifiers and implications prior to $\{G \vdash M : A\}^{@^i}$.

Case: $F = \{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n$

If $\Upsilon[@^i \hookleftarrow m] \vDash \{G \vdash M : A\}^{*^i}$ $\texttt{valid}$ did not hold, then by the definition it is clear that $\Upsilon[@^i \hookleftarrow m] \vDash \{G \vdash M : A\}^{*^i} \supset \ldots \supset F_n$ $\texttt{valid}$ would hold, so suppose that it does hold. Then there is a derivation for $G \vdash_\Sigma M \Leftarrow A$ which has some height $j < m$ and thus $\Upsilon[@^i \hookleftarrow j] \vDash \{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n$ $\texttt{valid}$ holds by definition. Since $j < m$, we can extract from the assumptions that $\Upsilon[@^i \hookleftarrow j] \vDash F_{k+1} \supset \ldots \supset F_n$ $\texttt{valid}$ holds. But $@^i$ and $*^i$ cannot appear in $F_{k+1} \supset \ldots \supset F_n$, so by Lemma 4.12 this judgement also holds for the height assignment $\Upsilon[@^i \hookleftarrow m]$. Therefore $\Upsilon[@^i \hookleftarrow m] \vDash \{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n$ $\texttt{valid}$ holds, as needed.

Case: $F = \forall x : \alpha.\mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$

Let ${F'}^{@^i}$ denote the body of the universal in F and ${F'}^{*^i}$ the same formula with $@^i$ replaced by $*^i$. Consider an arbitrary t such that $\mathcal{N} \cup \Theta_0 \vdash_{at} t : \alpha$. By definition, for any such t it must be that $\Upsilon[@^i \hookleftarrow l] \vDash {F'}^{@^i}[\![\{\langle x, t, \alpha\rangle\}]\!]$ $\texttt{valid}$ holds for any $l < m$. Thus by induction $\Upsilon[@^i \hookleftarrow m] \vDash {F'}^{*^i}[\![\{\langle x, t, \alpha\rangle\}]\!]$ $\texttt{valid}$ will hold. Therefore $\Upsilon[@^i \hookleftarrow m] \vDash \forall x : \alpha.{F'}^{*^i}$ $\texttt{valid}$ holds by definition.

Case: $F = \Pi\,\Gamma : \mathcal{C}.\mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$

Let ${F'}^{@^i}$ denote the body of the context quantification in F and ${F'}^{*^i}$ the same formula with

$@^i$ replaced by $*^i$. Consider an arbitrary context expression G such that $\mathcal{N};\emptyset \vdash \mathcal{C} \leadsto_{cs} G$ is derivable. By definition, for every $l < m$, $\Upsilon[@^i \hookleftarrow l] \vDash F'^{@^i}[G/\Gamma]$ `valid` will hold. Thus by induction, $\Upsilon[@^i \hookleftarrow m] \vDash F'^{*^i}[G/\Gamma]$ `valid` will hold. Therefore by Definition 4.26, $\Upsilon[@^i \hookleftarrow m] \vDash \Pi\,\Gamma : \mathcal{C}.F'^{*^i}$ `valid` must hold, as needed.

Case: $F = F_0 \supset \mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$

Let $F'^{@^i}$ denote the conclusion of the top-level implication in F and F'^{*^i} the same formula with $@^i$ replaced by $*^i$. Suppose that $\Upsilon[@^i \hookleftarrow m] \vDash F_0$ `valid` holds. Since $@^i$ and $*^i$ cannot appear in F_0, Lemma 4.12 permits us to conclude that $\Upsilon[@^i \hookleftarrow l] \vDash F_0$ `valid` holds for any l, in particular it must hold for any $l < m$. From this we can conclude using the assumptions that for any $l < m$, $\Upsilon[@^i \hookleftarrow l] \vDash F'^{@^i}$ `valid` holds, and thus by induction $\Upsilon[@^i \hookleftarrow m] \vDash F'^{*^i}$ `valid` will hold. Therefore $\Upsilon[@^i \hookleftarrow m] \vDash F_0 \supset F'^{*^i}$ `valid` will hold, as needed. □

Lemma 4.14. *Let F be a formula of the form*

$$\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))$$

in which the annotations $@^i$ or $*^i$ do not occur. Then $\Upsilon \vDash F$ `valid` holds if

$$\Upsilon[@^i \hookleftarrow m] \vDash \mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))\ \texttt{valid}.$$

holds for every natural number m.

Proof. We will prove this by induction on the formation of F, measuring size by the number of quantifiers and implications prior to $\{G \vdash M : A\}^{@^i}$.

Case: $F = \{G \vdash M : A\} \supset \ldots \supset F_n$

Suppose that $\Upsilon[@^i \hookleftarrow m] \vDash \{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n$ `valid` holds for any m. If the judgement $\Upsilon \vDash \{G \vdash M : A\}$ `valid` did not hold then $\Upsilon \vDash F$ `valid` holds vacuously so suppose it does hold. Then there must be a derivation of $G \vdash_\Sigma M \Leftarrow A$ and this derivation will have some height l. Therefore $\Upsilon[@^i \hookleftarrow l] \vDash \{G \vdash M : A\}^{@^i}$ `valid` will hold. So by the assumption, $\Upsilon[@^i \hookleftarrow l] \vDash F_{k+1} \supset \ldots \supset F_n$ `valid` must hold as well. Since $@^i$ does not appear in this formula, Lemma 4.12 permits us to conclude that $\Upsilon \vDash F_{k+1} \supset \ldots \supset F_n$ `valid` will hold, and therefore we can conclude $\Upsilon \vDash F$ `valid` holds, as needed.

Case: $F = \forall x : \alpha.\mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))$

Let F' denote the body of the universal in F and ${F'}^{@^i}$ similarly for the annotated formula. For an arbitrary term t such that $\mathcal{N} \cup \Theta_0 \vdash_{at} t : \alpha$ is derivable, we know by the assumptions that $\Upsilon[@^i \hookleftarrow m] \vDash {F'}^{@^i}[\![\{\langle x, t, \alpha\rangle\}]\!]$ `valid` holds for every m. By induction then, we can conclude that $\Upsilon \vDash F'[\![\{\langle x, t, \alpha\rangle\}]\!]$ `valid` must also hold. But then clearly $\Upsilon \vDash F$ `valid` holds by Definition 4.26.

Case: $F = \Pi\,\Gamma : \mathcal{C}.\mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))$

Let F' denote the body of the context quantification in F and ${F'}^{@^i}$ similarly for the annotated formula. For any context expression G such that $\mathcal{N}; \emptyset \vdash \mathcal{C} \rightsquigarrow_{cs} G$, we know by the assumptions that $\Upsilon[@^i \hookleftarrow m] \vDash {F'}^{@^i}[\![G/\Gamma]\!]$ `valid` holds for every m. We can conclude from this that $\Upsilon \vDash F'[G/\Gamma]$ `valid` will hold by an application of the inductive hypothesis. But then clearly $\Upsilon \vDash F$ `valid` holds by Definition 4.26.

Case: $F = F_0 \supset \mathcal{Q}'_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n)))$

Let F' denote the conclusion of the top-level implication in F and ${F'}^{@^i}$ similarly for the annotated formula. If $\Upsilon \vDash F_0$ `valid` were not valid, $\Upsilon \vDash F$ `valid` would be vacuously valid so suppose it were valid. Then by Lemma 4.12, it must be that $\Upsilon[@^i \hookleftarrow m] \vDash F_0$ `valid` holds for every m. By the assumptions then, $\Upsilon[@^i \hookleftarrow m] \vDash {F'}^{@^i}$ `valid` holds for every natural number m. An application of the inductive hypothesis permits us to conclude from this that $\Upsilon \vDash F'$ `valid` holds, and therefore that $\Upsilon \vDash F$ `valid` holds, as needed. □

Theorem 4.25. *If the premise sequent of an instance of the ind rule is valid and the requirement of non-occurrence of the annotations $@^i$ and $*^i$ is satisfied, then the conclusion sequent of the rule instance must be valid.*

Proof. In this proof we will use F to denote the formula

$$\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\} \supset \ldots \supset F_n))),$$

$F^{@^i}$ to denote $\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{@^i} \supset \ldots \supset F_n)))$ and F^{*^i} to denote $\mathcal{Q}_1.(F_1 \supset \ldots \supset \mathcal{Q}_{k-1}.(F_{k-1} \supset \mathcal{Q}_k.(\{G \vdash M : A\}^{*^i} \supset \ldots \supset F_n)))$. Let $\mathcal{S}$ denote the conclusion sequent, Ω the assumption formulas of $\mathcal{S}$, and $\mathcal{S}'$ the premise sequent. Consider an arbitrary height assignment Υ and closed instance of $\mathcal{S}$ identified by θ and σ. If any formula in $F' \in \Omega[\![\theta]\!][\sigma]$ were such that $\Upsilon \vDash F'$ `valid` did not hold then this closed instance of $\mathcal{S}$ would be vacuously valid with respect to Υ so suppose they all hold. Note that this θ and σ must also identify a closed instance of $\mathcal{S}'$ given that these sequents share support sets, term variable contexts, and context variable contexts. Since $@^i$ and $*^i$ do not occur in $\mathcal{S}$ by assumption, Lemma 4.12 permits us to conclude that for every m, $\Upsilon[@^i \hookleftarrow m] \vDash F'$ `valid` must hold. We will use this observation and the validity of the premise sequent to argue by strong induction on m that for all m, $\Upsilon[@^i \hookleftarrow m] \vDash F^{@^i}[\![\theta]\!][\sigma]$ `valid` holds. Once we have concluded that this holds for every m, Lemma 4.14 permits us to conclude that $\Upsilon \vDash F[\![\theta]\!][\sigma]$ `valid` holds and from this we conclude $\mathcal{S}[\![\theta]\!]_\emptyset[\sigma]$ is valid with respect to Υ.

Suppose that for all $l < m$, $\Upsilon[@^i \hookleftarrow l] \vDash F^{@^i}[\![\theta]\!][\sigma]$ `valid` holds. Then by Lemma 4.13 we easily conclude $\Upsilon[@^i \hookleftarrow m] \vDash F^{*^i}[\![\theta]\!][\sigma]$ `valid` holds. But then every assumption formula F'' of the closed sequent $\mathcal{S}'[\![\theta]\!]_\emptyset[\sigma]$ is such that $\Upsilon[@^i \hookleftarrow m] \vDash F''$ `valid` holds and therefore $\Upsilon[@^i \hookleftarrow m] \vDash F^{@^i}[\![\theta]\!][\sigma]$ `valid` will hold by the validity of $\mathcal{S}'$. Thus we can conclude that $\mathcal{S}$ is valid using Lemma 4.14 as described above. □

4.4.3 Additional Proof Rules that Interpret Annotations

We now take up the task of describing additional proof rules that take the meanings of annotations in formulas into consideration. These rules are an essential part of our proof system: without them, it would be impossible to construct proofs for the premises of instances of the induction rule.

One of the rules that we consider in this context is an enhanced version of the *id* rule. The rule that is included in the proof system currently requires the conclusion formula to be equi-valid to one of the assumption formulas. This requirement can be weakened with the refinement of the semantics that accommodates annotations in formulas. For example, if we have the formula $\{G \vdash M : A\}^*$ as an assumption, this suffices to ensure the validity of a sequent in which $\{G \vdash M : A\}^@$ is the conclusion formula. Similarly, the validity of either annotated formula would imply the validity of the unannotated atomic formula $\{G \vdash M : A\}$. This observation can be expanded to include non-atomic formulas with the proviso that the polarity of the occurrence of the formula must be paid attention to. For example, it is the validity of $\{G \vdash M : A\}^@ \supset F$ that implies that $\{G \vdash M : A\}^* \supset F$ is valid, rather than the other way around, and, further, the validity of both of these forms is implied by $\{G \vdash M : A\} \supset F$.

We formalize the idea discussed above through a notion of comparative strengths of formulas.

Definition 4.28 (Comparative Strengths of Annotated Formulas)**.** An atomic formula $\{G \vdash M : A\}^{Ann}$ is stronger than $\{G' \vdash M' : A'\}^{Ann'}$ with respect to Ξ and π if it is the case that $\Xi \vdash \{G \vdash M : A\} \equiv_\pi \{G' \vdash M' : A'\}$ and either $Ann = *^i$ and $Ann' = @^i$ or Ann' is no annotation and Ann is $@^i$, $*^i$, or no annotation. An implication formula $F_2 \supset F_2'$ is stronger than $F_1 \supset F_1'$ with respect to Ξ and π if F_1 is stronger than F_2 with respect to Ξ and π^{-1} and F_2' is stronger then F_1' with respect to Ξ and π. For any other arbitrary formula, F_2 is stronger than F_1 with respect to Ξ and π if their components satisfy the same relation, under a possibly extended Ξ in the case of context quantification, allowing for renaming of variables bound by quantifiers. The stronger than relation for formulas is represented by the judgement $\Xi \vdash F_2 \succeq_\pi F_1$.

The key result we show for this definition is that for any well-formed closed formula, we can indeed conclude that the validity of the stronger formula will ensure the validity of the other. In showing this we will rely on two substitution properties which are analogous to the Lemmas 4.2 and 4.3 about formula equivalence. We first prove the substitution properties and then use this lemma in proving the desired result about the stronger than relation.

Lemma 4.15. *Suppose that for some formulas F_1 and F_2, $\Xi \vdash F_2 \succeq_\pi F_1$ holds. Then both of the following properties hold.*

- *If θ is a hereditary substitution such that supp $(\theta) \cap$ supp $(\pi) = \emptyset$ and both $F_2[\![\theta]\!] = F_2'$ and $F_1[\![\theta]\!] = F_1'$ have derivations for some F_1' and F_2', then $\Xi[\![\theta]\!] \vdash F_2[\![\theta]\!] \succeq_\pi F_1[\![\theta]\!]$ holds.*

- *If σ is an appropriate substitution for Ξ with respect to some Ψ then the judgement $\Xi_\sigma \vdash F_2[\sigma] \succeq_\pi F_1[\sigma]$ holds.*

Proof. We observe that the application of both term and context variable substitutions to formulas does not impact on either their structure or their annotations. Therefore we can conclude that both of these clauses hold through an inductive argument on the formation of both F_2 and F_1. □

Theorem 4.26. *For a height assignment Υ and well-formed closed formulas F_1 and F_2, if $\Xi \vdash F_2 \succeq_\pi F_1$ and F_2 is valid with respect to Υ then F_1 is valid with respect to Υ.*

Proof. We prove this by induction on the formation of F_1 and F_2. Suppose that F_2 is valid with respect to Υ, and consider the possible structures for F_1 and F_2.

Case: $F_1 = \{G_1 \vdash M_1 : A_1\}^{Ann_1}$ **and** $F_2 = \{G_2 \vdash M_2 : A_2\}^{Ann_2}$

We first observe that since $\Xi \vdash \{G_2 \vdash M_2 : A_2\} \equiv_\pi \{G_1 \vdash M_1 : A_1\}$, we can extract from this that $\pi.(\{G_2 \vdash M_2 : A_2\}) = \{G_1 \vdash M_1 : A_1\}$ and thus the unannotated forms of these formulas will be equi-valid with respect to Υ. So by the assumption of validity for F_2, we can determine from this that $\vdash_\Sigma G_1$ `ctx` and $G_1 \vdash_\Sigma A_1$ `type` have derivations, and also that $G_1 \vdash_\Sigma M_1 \Leftarrow A_1$ has a derivation which satisfies the annotation Ann_2 with respect to Υ. The only remaining piece is then to show that this derivation for $G_1 \vdash_\Sigma M_1 \Leftarrow A_1$ will also satisfy the annotation Ann_1 with respect to Υ. There are two possibilities to consider, either $Ann_2 = *^i$ and $Ann_1 = @^i$ or Ann_1 is no annotation and Ann_2 is $@^i$, $*^i$, or no annotation. If $Ann_2 = *^i$ it is clear that if a derivation of height strictly less than $\Upsilon(@^i)$ exists then this same derivation will also be of a height less than or equal to $\Upsilon(@^i)$. In the other case, regardless of the height of the derivation which satisfies Ann_2 this same derivation will be sufficient to determine that the unannotated form of the formula will be valid. Therefore F_1 must be valid with respect to Υ.

$$\frac{F_2 \in \Omega \quad supp(\pi) \subseteq \mathbb{N} \quad \Xi \vdash F_2 \succeq_\pi F_1}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow F_1}\ id$$

Figure 4.7: Id Rule allowing Annotated Formulas

Case: $F_1 = F_1' \supset F_1''$ and $F_2 = F_2' \supset F_2''$

If F_1' were not valid with respect to Υ, then F_1 must clearly be valid with respect to Υ. If F_1' were valid with respect to Υ, then by an application of the inductive hypothesis F_2' must be valid with respect to Υ. Thus by the validity of F_2, F_2'' must be valid with respect to this same height assignment, and so by a second application of the inductive hypothesis F_1'' is valid with respect to Υ, as needed.

Case: F_1 and F_2 are of some other structure

For the remaining cases the desired result is a direct result of the definition of validity with respect to a height assignment Υ and an application of the inductive hypothesis on the components of the formulas, using Lemma 4.15 to address the instantiation of quantifiers in the relevant cases. □

The new version of the identity rule is presented in Figure 4.7. Figure 4.8 presents enhancements to the atomic proof rules that also take into account the semantics of annotations. As usual, we show that these rules preserve the well-formedness of sequents and are also sound.

Theorem 4.27. *The following properties holds for every instance of each of the rules in Figures 4.7 and 4.8:*

1. *If the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then all the sequent premises must be well-formed.*

2. *If the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and the premise sequent is valid, then the conclusion sequent must also be valid.*

$$\frac{\begin{array}{c} Ann \in \{*^i, @^i\} \\ \mathcal{CS} = \mathit{AllCases}\,(\mathbb{N};\Psi;\Xi;\Omega,\{G \vdash R : P\} \longrightarrow F, \{G \vdash R : P\}) \\ \left\{ \begin{array}{l} \mathbb{N}';\Psi';\Xi';\Omega',\{G_1 \vdash M_1 : A_1\}^{*^i},\ldots,\{G_k \vdash M_k : A_k\}^{*^i} \longrightarrow F' \\ \quad \mid\ \mathbb{N}';\Psi';\Xi';\Omega',\{G_1 \vdash M_1 : A_1\},\ldots,\{G_k \vdash M_k : A_k\} \longrightarrow F' \in \mathcal{CS} \end{array} \right\} \end{array}}{\mathbb{N};\Psi;\Xi;\Omega,\{G \vdash R : P\}^{Ann} \longrightarrow F}\ \textit{atm-app-L}$$

$$\frac{\begin{array}{c} h : \Pi x_1{:}A_1.\ \ldots \Pi x_n{:}A_n.\, P \in \Sigma \text{ or the explicit bindings in } G \\ \{G \vdash N : B\} \in \Omega \qquad P[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_n, M_n, (A_n)^-\rangle\}]\!] = P' \\ \left\{ \begin{array}{l} \mathbb{N};\Psi;\Xi;\Omega \longrightarrow \\ \quad \left\{G \vdash M_i : A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!]\right\}^{*^i} \\ \hfill \mid\ 1 \le i \le n \end{array} \right\} \end{array}}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \{G \vdash h\ M_1 \ldots M_n : P'\}^{@^i}}\ \textit{atm-app-R}$$

$$\frac{\begin{array}{l} \qquad\qquad n \notin \mathit{dom}\,(\mathbb{N}) \quad Ann \in \{*^i, @^i\} \\ \Xi' = \left\{ \begin{array}{ll} (\Xi \setminus \{\Gamma{\uparrow}\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}]\}) \cup \left\{\Gamma{\uparrow}(\mathbb{N}_\Gamma, n : (A_1)^-) : \mathcal{C}[\mathcal{G}]\right\} & \text{if } \Gamma \text{ in } G \\ \Xi & \text{otherwise} \end{array} \right. \\ \mathbb{N}, n : (A_1)^-;\Psi;\Xi'; \\ \quad \Omega, \left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^-\rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\right\}^{*^i} \longrightarrow F \end{array}}{\mathbb{N};\Psi;\Xi;\Omega,\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}^{Ann} \longrightarrow F}\ \textit{atm-abs-L}$$

$$\frac{\begin{array}{l} \qquad\qquad n \notin \mathit{dom}\,(\mathbb{N}) \\ \Xi' = \left\{ \begin{array}{ll} (\Xi \setminus \{\Gamma{\uparrow}\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}]\}) \cup \left\{\Gamma{\uparrow}(\mathbb{N}_\Gamma, n : (A_1)^-) : \mathcal{C}[\mathcal{G}]\right\} & \text{if } \Gamma \text{ in } G \\ \Xi & \text{otherwise} \end{array} \right. \\ \mathbb{N}, n : (A_1)^-;\Psi;\Xi';\Omega \longrightarrow \\ \quad \left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^-\rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\right\}^{*^i} \end{array}}{\mathbb{N};\Psi;\Xi;\Omega \longrightarrow \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}^{@^i}}\ \textit{atm-abs-R}$$

Figure 4.8: Atomic Proof Rules allowing Annotated Formulas

Proof. Given that the well-formedness of these sequents is defined by the well-formedness of the sequent with annotations erased, clause (1) holds trivially by Theorems 4.15 and 4.21. For the second clause, consider each of the rules in Figures 4.7 and 4.8.

Case: *id*

Consider an arbitrary height assignment Υ and a closed instance of the conclusion sequent identified by θ and σ such that $supp(\theta) \cap supp(\pi) = \emptyset$. Note that a proof for such a restricted case will be sufficient to conclude the validity for every closed instance via Theorem 4.11. The requirements on θ and σ will clearly be sufficient to ensure that they satisfy the conditions of Lemma 4.15 and therefore from the the assumptions we can conclude that judgement $\emptyset \vdash F_2[\![\theta]\!][\sigma] \succeq_\pi F_1[\![\theta]\!][\sigma]$ holds. If any formula in $\Omega[\![\theta]\!][\sigma]$ were not valid with respect to Υ then this closed instance would be vacuously valid so suppose they are all valid. Then in particular, $F_2[\![\theta]\!][\sigma]$ is valid with respect to Υ. Therefore by Theorem 4.26 it must be that $F_1[\![\theta]\!][\sigma]$ is also valid, as needed.

Case: *atm-app-L*

This argument follows that given for Theorem 4.19 generalized over an arbitrary height assignment Υ for any annotations appearing in the conclusion sequent. The only significant change is to observe in the application of Lemma 4.10 that whenever there is a derivation for $G[\![\theta]\!][\sigma] \vdash_\Sigma R[\![\theta]\!] \Leftarrow P[\![\theta]\!]$ of some height k, it will be the case that for each i, $1 \leq i \leq n$,

$$G[\![\theta]\!][\sigma] \vdash_\Sigma M_i \Leftarrow A_i[\![\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^- \rangle]\!]$$

has a derivation of height smaller than k. Thus it is sound to annotate the formulas $\big\{\big\{G[\![\theta]\!][\sigma] \vdash M_i : \big(A_i[\![\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^- \rangle]\!]\big)\big\}\big|1 \leq i \leq n\big\}$ with $*^i$ as it is known that they must be derivable with a smaller height and thus will satisfy the requirements of this annotation for the height assignment.

Case: *atm-app-R*

Let A'_i denote the type $A_i[\![\{\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^- \rangle\}]\!]$. Consider an arbitrary height annotation Υ and closed instance identified by θ and σ. Suppose all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ, since if they were not this instance would be vacuously valid. Then since clearly Υ will be a valid height assignment for all of the premise sequents, and also θ and σ will identify closed instances of these sequents, by the validity of the premise sequents all of the formulas $\{G \vdash M_i : A'_i\}^{*^i}[\![\theta]\!][\sigma]$ are valid with respect to Υ. But then clearly we can ensure that the formula $\{G \vdash h\ M_1 \ldots M_n : P\}^{@^i}[\![\theta]\!][\sigma]$ is valid by applying Theorem 2.12 with the LF derivations for $G[\![\theta]\!][\sigma] \vdash_\Sigma M_i[\![\theta]\!] \Leftarrow A'_i[\![\theta]\!]$ which must have heights smaller than the height assigned to $@^i$ by Υ.

Case: *atm-abs-L*

Consider for the conclusion sequent an arbitrary height assignment Υ and closed instance identified by θ and σ. Assume that n is a name which does not appear in θ or σ; as Theorem 4.11 would permit permuting this name to one which does not. Suppose all formulas in $(\Omega, \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}^{Ann})[\![\theta]\!][\sigma]$ are valid with respect to Υ, as otherwise this instance would be vacuously valid. In particular then, the closed formula $\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}^{Ann}[\![\theta]\!][\sigma]$ is valid with respect to Υ. Thus letting k be the height assigned to $@^i$ by Υ there are derivations of $\vdash_\Sigma G[\![\theta]\!][\sigma]\ \texttt{ctx}$, $G[\![\theta]\!][\sigma] \vdash_\Sigma \Pi x{:}A_1.\, A_2[\![\theta]\!]\ \texttt{type}$, and $G[\![\theta]\!][\sigma] \vdash_\Sigma \lambda x.\, M[\![\theta]\!] \Leftarrow \Pi x{:}A_1.\, A_2[\![\theta]\!]$ of height k. But then clearly from subderivations of these and an application of *CTX_TERM* there would be derivations shorter than k for the LF judgements $\vdash_\Sigma (G, n : A_1)[\![\theta]\!][\sigma]\ \texttt{ctx}$, $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!][\![\theta]\!]\ \texttt{type}$, as well as for $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\{\langle x, n, (A_1)^- \rangle\}]\!][\![\theta]\!] \Leftarrow A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!][\![\theta]\!]$. Thus all the formulas in $(\Omega, \left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^- \rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!]\right\}^{*^i})[\![\theta]\!][\sigma]$ are valid and so by the validity of the premise sequents $F[\![\theta]\!][\sigma]$ will be valid, as needed.

Case: *atm-abs-R*

Consider for the conclusion sequent an arbitrary height assignment Υ and closed instance identified by θ and σ. Assume that n is a name which does not appear in θ or σ; as Theorem 4.11 would permit permuting this name to one which does not. Suppose all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ, as otherwise this instance would be vacuously valid. Since θ and σ also identify a closed instance of the premise sequent and all formulas in $\Omega[\![\theta]\!][\sigma]$ are valid, we can infer from the validity of the premise sequent that $\left\{G, n : A_1 \vdash M[\![\{\langle x, n, (A_1)^- \rangle\}]\!] : A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!]\right\}^{*^i}[\![\theta]\!][\sigma]$ is valid with respect to Υ. So by definition there exist LF derivations for

1. $\vdash_\Sigma (G, n : A_1)[\![\theta]\!][\sigma]$ `ctx`,

2. $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!]$ `type`, and

3. $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!] \Leftarrow A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!][\![\theta]\!]$

of height less than that assigned to $@^i$ by Υ. But then clearly we can construct derivations also for the LF judgements $\vdash_\Sigma G[\![\theta]\!][\sigma]$ `ctx`, $G[\![\theta]\!][\sigma] \vdash_\Sigma \Pi x{:}A_1.\, A_2[\![\theta]\!]$ `type`, and $G[\![\theta]\!][\sigma] \vdash_\Sigma \lambda x.\, M[\![\theta]\!] \Leftarrow \Pi x{:}A_1.\, A_2[\![\theta]\!]$ of a height satisfying the annotation assigned to $@^i$ by Υ. Therefore the formula $\{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}^{@^i}[\![\theta]\!][\sigma]$ will be valid with respect to Υ, as needed. □

4.5 Proof Rules Encoding LF Meta-Theorems

The meta-theorems concerning LF derivability that were discussed in Section 2.3 are often useful in informal arguments about the properties of LF specifications. In this section, we describe proof rules that provide a means for using these meta-theorems in formal reasoning based on our logic.

The weakening meta-theorem has a proviso that the type for the new binding introduced into the context must be well-formed. This must be reflected in the proof rule that captures the content of this meta-theorem by a collection of premises that check that this property of the type will hold. We refer to the process that generates the typing judgements that must be checked towards this end as *type decomposition*. In addition to the type, this process is parameterized by a collection of nominal constants, a context variable context and a context expression. The result of type decomposition is a collection of triples that comprise an extended collection of nominal constants, a modified context variable context and an atomic formula expressing a typing judgement that must be checked. This idea is made precise below.

Definition 4.29 (Decomposition of Types)**.** The decomposition of a canonical type A with respect to a collection of nominal constants $\mathbb{N}$, a context variable context Ξ and a context expression G, notated as *Decompose* $(\mathbb{N}; \Xi; G; A)$, is defined as follows:

1. If A is a type of the form $(a\ M_1 \ldots M_n)$ where $a : \Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n.\, \mathrm{Type} \in \Sigma$, then $\mathit{Decompose}\,(\mathbb{N};\Xi;G;A)$ is the collection

$$\bigcup_{i \in 1..n} \left\{ \left(\mathbb{N}, \Xi, \left\{ G \vdash M_i : (A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!])\right\}\right)\right\}.$$

2. If $A = \Pi x{:}A_1.\, A_2$, then letting G' be $G, n : A_1$, $\mathbb{N}'$ be $\mathbb{N} \cup \{n\}$, and Ξ' be the set

$$\begin{aligned}\{\ \Gamma{\uparrow}\mathbb{N}'_\Gamma : \mathcal{C}[\mathcal{G}] \mid \Gamma{\uparrow}\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi \text{ and} \\ \mathbb{N}'_\Gamma \text{ is } \mathbb{N}_\Gamma \cup \{n\} \ \text{ if } \Gamma \text{ occurs in } G \text{ and } \mathbb{N}_\Gamma \ \text{ otherwise } \}\end{aligned}$$

for some nominal constant $n : (A_1)^- \in \mathcal{N} \setminus \mathbb{N}$, $\mathit{Decompose}\,(\mathbb{N};\Xi;G;A)$ is the collection $\mathit{Decompose}\,(\mathbb{N};\Xi;G;A_1) \cup \mathit{Decompose}\left(\mathbb{N}';\Xi';G';A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\right)$.

Note that the decomposition will not always be defined, but as we will show, it will be defined in all the cases we need to use it. Further, we will wish to be careful in how the name n in the second clause of this definition is selected in practice for this decomposition to be useful in reasoning. We will show that the way this definition is used will be sound regardless of the choice of name, however choices for n which are not fresh will lead to generating formulas which are never provable and thus we will wish to avoid these name in an implementation of the proof system.

Figure 4.9 presents the proof rules which encode the content of the LF meta-theorems. Note in particular that the goal formula of the conclusion sequent in each of these rules expresses the meta-theorem in terms of the atomic formulas in the logic. The symbol Ann in the first three rules, which encode weakening, strengthening, and context permutation, stands for no annotation, $@^i$ or $*^i$ for some i, used in the same manner throughout the rule instance. Permitting annotations in these rules is justified by the fact that the corresponding meta-theorems guarantee the preservation of the structure, and thus height, of LF derivations. Clearly instantiation does not share this property and so we do not consider annotated formulas for this proof rule.

As before we show that extending the proof system with these rules will maintain both the well-formedness and soundness properties. The following lemma, which ensures that

$$\frac{\begin{array}{c} n \text{ does not appear in } \; M, \; A, \text{ or the explicit bindings in } G \\ \text{if } \Gamma_i{\uparrow}\mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi \text{ and } \Gamma_i \text{ appears in } G, \text{ then } n \in \mathbb{N}_i \\ \{\mathbb{N}'; \Psi; \Xi'; \Omega \longrightarrow F' \mid (\mathbb{N}', \Xi', F') \in \mathit{Decompose}(\mathbb{N}; \Xi; G; B)\} \end{array}}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow \{G \vdash M : A\}^{Ann} \supset \{G, n : B \vdash M : A\}^{Ann}} \; \textit{LF-wk}$$

$$\frac{n \text{ does not appear in } M, \, A, \text{ or the explicit bindings in } G}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow \{G, n : B \vdash M : A\}^{Ann} \supset \{G \vdash M : A\}^{Ann}} \; \textit{LF-str}$$

$$\frac{\begin{array}{c} G = G'', n_1 : A_1, n_2 : A_2, n_3 : A_3, \ldots, n_m : A_m \\ G' = G'', n_2 : A_2, n_1 : A_1, n_3 : A_3, \ldots, n_m : A_m \\ n_1 \text{ does not appear in } A_2 \end{array}}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow \{G \vdash M : A\}^{Ann} \supset \{G' \vdash M : A\}^{Ann}} \; \textit{LF-perm}$$

$$\frac{\begin{array}{c} G' = n_1 : A_1, \ldots, n_m : A_m \qquad G'' = G, n_2 : A_2', \ldots, n_m : A_m' \\ M[\![\{\langle n_1, M_1, (A_1)^- \rangle\}]\!] = M' \qquad A[\![\{\langle n_1, M_1, (A_1)^- \rangle\}]\!] = A' \\ \left\{A_i[\![\{\langle n_1, M_1, (A_1)^- \rangle\}]\!] = A_i' \;\mid\; 2 \leq i \leq m\right\} \end{array}}{\mathbb{N}; \Psi; \Xi; \Omega \longrightarrow \{G, G' \vdash M : A\} \supset \{G \vdash M_1 : A_1\} \supset \{G'' \vdash M' : A'\}} \; \textit{LF-inst}$$

Figure 4.9: Rules Encoding Meta-Theoretic Properties of LF

whenever the decomposition is performed on a well-formed type relative to well-formed contexts, the result is defined and further that the formulas will themselves be well-formed, will be useful in showing the well-formedness property for the weakening proof rule.

Lemma 4.16. *Assume* Ξ *is a context variable context such that for each* $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ *there is a derivation of* $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ *ctx-ty. Also assume* $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi \vdash G$ *context and* $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ *type have derivations. Then* $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ *is defined and for each* $(\mathbb{N}', \Xi', F) \in \mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ *it is the case that*

1. *for each* $\Gamma_i' \uparrow \mathbb{N}_i' : \mathcal{C}_i'[\mathcal{G}_i'] \in \Xi'$ *there is a derivation of* $\mathbb{N}' \setminus \mathbb{N}_i'; \Psi \vdash \mathcal{C}_i'[\mathcal{G}_i']$ *ctx-ty and*

2. F *is a well-formed formula with respect to* $(\mathbb{N}' \cup \Theta_0 \cup \Psi)$ *and* Ξ'^{-}*.*

Proof. We prove this by induction on the formation of the type B. Consider the cases for the structure of B.

Case: B is an atomic type P.

Given that $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ *type* is derivable, B is of the form $(a\ M_1 \ldots M_n)$ and there must exist $a : K$ in the LF signature for a kind of the form $\Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n.$ Type and subderivations of $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M_i : (A_i)^-$ for each M_i. From these observations $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ is clearly defined. Noting that the arity kinding for B will ensure the substitution application is defined, let $A_i[\![\{\langle x_1, M_1, (A_1)^- \rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^- \rangle\}]\!]$ be denoted by A_i' for each i. Then we can express the collection $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ as the set of tuples $\bigcup_{i \in 1..n} (\mathbb{N}, \Xi, \{G \vdash M_i : A_i'\})$.

Clearly, for any tuple in $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ the context variable context satisfies condition (1). Since $a : K$ is from the LF signature it is obvious that $\cdot \vdash_\Sigma K$ `type` is derivable in LF, and thus for each i, $1 \leq i \leq n$, $\{x_1 : A_1, \ldots, x_{i-1} : A_{i-1}\} \uplus (\mathbb{N} \cup \Theta_0 \cup \Psi) \vdash_{ak} A_i$ *type* is derivable. By Theorem 3.1 there must exist derivations of $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_i'$ *type* for each i. We can then conclude that there are derivations of $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi \vdash G$ *context*, $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_i'$ *type* and $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{at} M_i : (A_i)^-$, and therefore for each i, $1 \leq i \leq n$, $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi^- \vdash \{G \vdash M_i : A_i'\}$ *fmla* is derivable, satisfying condition (2).

Case: B is a canonical type $\Pi x{:}A_1.\, A_2$.

Then letting $G' = G, n : (A_1)^-$, $\mathbb{N}' = \mathbb{N} \cup \{n\}$, and

$$\Xi' = \{\ \Gamma{\uparrow}\mathbb{N}'_\Gamma : \mathcal{C}[\mathcal{G}] \mid \Gamma{\uparrow}\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi \text{ and } \mathbb{N}'_\Gamma \text{ is } \mathbb{N}_\Gamma \cup \{n\} \text{ if } \Gamma \text{ occurs in } G \text{ and } \mathbb{N}_\Gamma \text{ otherwise } \}$$

for some new nominal constant n, the decomposition *Decompose* $(\mathbb{N}; \Xi; G; B)$ is defined if both *Decompose* $(\mathbb{N}; \Xi; G; A_1)$ and *Decompose* $\big(\mathbb{N}'; \Xi'; G'; A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\big)$ are defined, and it will be the union of these two sets. Given that $\mathbb{N}\backslash\mathbb{N}_i = (\mathbb{N}, n : (A_1)^-)\backslash(\mathbb{N}_i, n : (A_1)^-)$ for any $\mathbb{N}_i$, the context variable context Ξ' will clearly satisfy the requirements of this lemma. From the derivation of $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ *type* we can obtain derivations of both $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_1$ *type* and $\{x : (A_1)^-\} \uplus \mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_2$ *type*. By Theorem 3.1 there must then exist a derivation for $\mathbb{N}' \cup \Theta_0 \cup \Psi \vdash_{ak} A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]$ *type*. From the derivations of $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi \vdash G$ *context* and $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_1$ *type* we can construct a derivation for the judgement $\mathbb{N}' \cup \Theta_0 \cup \Psi; \Xi \vdash G, n : A_1$ *context*. The type A_1 is clearly smaller than B, and it is straightforward to conclude that $A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]$ must be as well. Thus by invoking the inductive hypothesis twice we determine that both *Decompose* $(\mathbb{N}; \Xi; G; A_1)$ and *Decompose* $\big(\mathbb{N}'; \Xi'; G'; A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\big)$ are defined, and that each tuple in these sets satisfy the conditions (1) & (2). Therefore *Decompose* $(\mathbb{N}; \Xi; G; B)$ will be defined and each tuple in this set will satisfy the necessary conditions. □

We now show the well-formedness property for the rules in Figure 4.9. The only interesting case to consider is for *LF-wk*.

Theorem 4.28. *The following property holds of the rules in Figure 4.9: if the conclusion sequent is well-formed, the premises expressing typing conditions have derivations and the conditions expressed by the other, non-sequent premises are satisfied, then the premise sequents must be well-formed.*

Proof. Consider each of the rules defined in Figure 4.9.

Case: ***LF-str*, *LF-perm*, and *LF-inst*.**

There are no premise sequents in these rules so the property holds vacuously.

Case: *LF-wk*

Given the well-formedness of the conclusion sequent, $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty is derivable for each $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ and $\mathbb{N} \cup \Psi \cup \Theta_0; \Xi^- \vdash F$ *fmla* is derivable for each formula $F \in \Omega \cup \{\{G \vdash M : A\}^{Ann} \supset \{G, n : B \vdash M : A\}^{Ann}\}$. From this it is obvious that G is a well-formed context expression with respect to $(\mathbb{N} \cup \Psi \cup \Theta_0)$ and Ξ, and that B is a good type with respect to $(\mathbb{N} \cup \Psi \cup \Theta_0)$. Thus by Lemma 4.16 the decomposition is defined and for each $(\mathbb{N}', \Xi', F) \in \mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ the following hold

1. for each $\Gamma'_i \uparrow \mathbb{N}'_i : \mathcal{C}'_i[\mathcal{G}'_i] \in \Xi'$, $\mathbb{N}' \setminus \mathbb{N}'_i; \Psi \vdash \mathcal{C}'_i[\mathcal{G}'_i]$ ctx-ty has a derivation and
2. $\mathbb{N}' \cup \Theta_0 \cup \Psi; \Xi'^- \vdash F$ *fmla* has a derivation.

By Theorem 3.2 we can determine that every formula in Ω is well-formed under the extended contexts $(\mathbb{N}' \cup \Psi \cup \Theta_0)$ and Ξ'^-, and thus all of the premise sequents will be well-formed. $\square$

The following lemma captures the intended meaning of the type decomposition; that the typing judgements identified by type decomposition are sufficient in determining that any instance of the type is well-formed in LF. Thus result will be key to proving soundness of the weakening proof rule.

Lemma 4.17. *Assume Ξ is a context variable context such that for each $\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i] \in \Xi$ there is a derivation of $\mathbb{N} \setminus \mathbb{N}_i; \Psi \vdash \mathcal{C}_i[\mathcal{G}_i]$ ctx-ty. Also assume $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi \vdash G$ context and $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ type have derivations. Let θ and σ be some substitutions such that dom $(\theta) \subseteq \Psi$, dom $(\sigma) \subseteq \Xi$, and for every $\mathbb{N}', \Xi', F' \in$ Decompose $(\mathbb{N}; \Xi; G; B)$ the formula $F'[\![\theta]\!][\sigma]$ is defined and valid, then $G[\![\theta]\!][\sigma] \vdash_\Sigma B[\![\theta]\!]$* `type` *is derivable in LF.*

Proof. We begin by observing that Lemma 4.16 ensures that the decomposition will be defined. The proof then proceeds by induction on the formation of B. Consider the possible structures for B.

Case: B is an atomic type P.

Given that $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ *type* is derivable, B is of the form $(a\ M_1 \ldots M_n)$ and there must exist $a : K$ in the LF signature for a kind of the form $\Pi x_1{:}A_1. \ldots \Pi x_n{:}A_n.$ Type.

Since the arity kinding for B will ensure the substitution application is defined, let A'_i denote the type $A_i[\![\{\langle x_1, M_1, (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}, (A_{i-1})^-\rangle\}]\!]$ for each i, $1 \leq i \leq n$. We can then express the collection $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ as $\bigcup_{i\in 1..n}(\mathbb{N}, \Xi, \{G \vdash M_i : A'_i\})$ given that it must be defined. If the formula $\{G \vdash M_i : A'_i\}\,[\![\theta]\!][\sigma]$ is defined and valid for each $(\mathbb{N}, \Xi, \{G \vdash M_i : A'_i\}) \in \mathit{Decompose}(\mathbb{N}; \Xi; G; B)$, then there must be derivations of $\vdash_\Sigma G[\![\theta]\!][\sigma]\ \mathtt{ctx}$, $G[\![\theta]\!][\sigma] \vdash_\Sigma A'_i[\![\theta]\!]\ \mathtt{type}$, and $G[\![\theta]\!][\sigma] \vdash_\Sigma M_i[\![\theta]\!] \Leftarrow A'_i[\![\theta]\!]$ for each i. By Theorem 2.3 the type $A_i[\![\theta]\!][\![\{\langle x_1, M_1[\![\theta]\!], (A_1)^-\rangle, \ldots, \langle x_{i-1}, M_{i-1}[\![\theta]\!], (A_{i-1})^-\rangle\}]\!]$ is the same as $A'_i[\![\theta]\!]$ since θ does not make substitution for any x_j. From these derivations and the fact that $a : K$ is in the LF signature, Theorem 2.13 ensures that there exists a derivation $G[\![\theta]\!][\sigma] \vdash_\Sigma B[\![\theta]\!] \Rightarrow \text{Type}$. From this we can easily infer the derivability of $G[\![\theta]\!][\sigma] \vdash_\Sigma B[\![\theta]\!]\ \mathtt{type}$, as is needed.

Case: B is a canonical type $\Pi x{:}A_1.\,A_2$.

Letting $G' = G, n : (A_1)^-$, $\mathbb{N}' = \mathbb{N} \cup \{n\}$, and

$$\begin{aligned} \Xi' = \{\ & \Gamma{\uparrow}\mathbb{N}'_\Gamma : \mathcal{C}[\mathcal{G}] \mid \Gamma{\uparrow}\mathbb{N}_\Gamma : \mathcal{C}[\mathcal{G}] \in \Xi \text{ and} \\ & \qquad \mathbb{N}'_\Gamma \text{ is } \mathbb{N}_\Gamma \cup \{n\} \ \text{ if } \Gamma \text{ occurs in } G \text{ and } \mathbb{N}_\Gamma \ \text{ otherwise } \} \end{aligned}$$

for a new nominal constant n, the decomposition $\mathit{Decompose}(\mathbb{N}; \Xi; G; B)$ is defined to be the set $\mathit{Decompose}(\mathbb{N}; \Xi; G; A_1) \cup \mathit{Decompose}\big(\mathbb{N}'; \Xi'; G'; A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]\big)$. Given that $\mathbb{N} \setminus \mathbb{N}_i = (\mathbb{N}, n : (A_1)^-) \setminus (\mathbb{N}_i, n : (A_1)^-)$ for any $\mathbb{N}_i$, the context variable context Ξ' will then satisfy the requirements of this lemma. From the derivation of $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} B$ *type* we can extract derivations for $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_1$ *type* and $\{x : (A_1)^-\} \uplus (\mathbb{N} \cup \Theta_0 \cup \Psi) \vdash_{ak} A_2$ *type*. By Theorem 3.1 $\mathbb{N}' \cup \Theta_0 \cup \Psi \vdash_{ak} A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]$ *type* must then be derivable. We have as assumptions derivations for both $\mathbb{N} \cup \Theta_0 \cup \Psi; \Xi \vdash G$ *context* and $\mathbb{N} \cup \Theta_0 \cup \Psi \vdash_{ak} A_1$ *type*, thus we are able to construct a derivation for $\mathbb{N}' \cup \Theta_0 \cup \Psi; \Xi \vdash G, n : A_1$ *context*. The type A_1 is clearly smaller than B, and it is easy to argue that $A_2[\![\{\langle x, n, (A_1)^-\rangle\}]\!]$ must be as well. Thus by invoking the inductive hypothesis twice we determine that both $G[\![\theta]\!][\sigma] \vdash_\Sigma A_1[\![\theta]\!]\ \mathtt{type}$ and $(G, n : A_1)[\![\theta]\!][\sigma] \vdash_\Sigma A'_2[\![\theta]\!]\ \mathtt{type}$ are derivable. From these we can construct a derivation for $G[\![\theta]\!][\sigma] \vdash_\Sigma \Pi x{:}A_1.\,A_2[\![\theta]\!]\ \mathtt{type}$ using an application of *CANON_FAM_PI*. $\square$

We conclude this section by proving the soundness of these proof rules with respect to the semantics.

Theorem 4.29. *The following property holds for every instance of each of the rules in Figure 4.9: if the premises expressing typing judgements are derivable, the conditions described in the other non-sequent premises are satisfied and all the premise sequents are valid, then the conclusion sequent must also be valid.*

Proof. Consider each rule in Figure 4.9.

Case: *LF-wk*

Let Υ be an arbitrary height assignment and θ and σ identify an arbitrary closed instance of the conclusion sequent. Suppose that all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ as if any were not this closed instance would be vacuously valid. The instance would be similarly valid if $\{G \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ were not valid, so suppose it is. Further, assume that any nominal constants in $\mathbb{N}'\backslash\mathbb{N}$ do not appear in θ or σ as by Theorem 4.11 the validity of any instance which does use such nominal constants is ensured by the validity of every instance which does not. Such θ and σ will also identify closed instances for each of the premise sequents. From the validity of these premise sequents, we can conclude by Lemma 4.17 that $G[\![\theta]\!][\sigma] \vdash_\Sigma B[\![\theta]\!]$ `type` is derivable in LF. By the validity of $\{G \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ there must exist LF derivations for

1. $\vdash_\Sigma G[\![\theta]\!][\sigma]$ `ctx`,
2. $G[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and
3. $G[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$

which satisfy the restriction on the derivation height identified by *Ann* with respect to Υ. Since n cannot appear in G, M, or A by assumption or in θ by virtue of its being substitution compatible with the conclusion sequent, we can construct a derivation for $\vdash_\Sigma (G, n : B)[\![\theta]\!][\sigma]$ `ctx` through an application of *CTX_TERM* and also conclude that $(G, n : B)[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and $(G, n : B)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$ have derivations of the same height as (2) and (3) respectively through the application of Theorem 2.8. Thus the restriction identified for the annotation Ann with respect to Υ will clearly also be satisfied by this derivation for the later judgement, and so $\{G, n : B \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ will be valid with respect to Υ. Thus this closed instance of the sequent must be valid.

Case: *LF-str*

Let Υ be an arbitrary height assignment and θ and σ identify an arbitrary closed instance of the conclusion sequent. Suppose that all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ as if any were not this closed instance would be vacuously valid. Consider the goal formula $\{G, n : B \vdash M : A\}^{Ann} \supset \{G \vdash M : A\}^{Ann}$. If $\{G, n : B \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ were not valid with respect to Υ then this instance of the formula would be vacuously valid, so assume it is valid with respect to Υ. Then there must be derivations of

1. $\vdash_\Sigma (G, n : B)[\![\theta]\!][\sigma]$ `ctx`,
2. $(G, n : B)[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and
3. $(G, n : B)[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$

of some height m which satisfies the height restriction identified by Ann with respect to Υ. The derivation of (1) must conclude by *CTX_TERM* using a derivation of $\vdash_\Sigma G[\![\theta]\!][\sigma]$ `ctx` and n cannot appear in the context $G[\![\theta]\!][\sigma]$. Since n does not appear in A or M, and since it cannot be in the support of the substitution θ, it cannot appear in $M[\![\theta]\!]$ or $A[\![\theta]\!]$. So by Theorem 2.9 the judgements $G[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and $G[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$ have derivations and the later judgement has a derivation of height m. Thus it clearly also satisfies the height restriction identified by Ann with respect to Υ. Therefore the formula $\{G \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ is valid with respect to Υ, as needed.

Case: *LF-perm*

Let Υ be an arbitrary height assignment and θ and σ identify an arbitrary closed instance of the conclusion sequent. Suppose that all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ, as if any were not this closed instance would be vacuously valid. Consider the goal formula under this instance, $(\{G \vdash M : A\}^{Ann} \supset \{G' \vdash M : A\}^{Ann})[\![\theta]\!][\sigma]$. If $\{G \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$

were not valid with respect to Υ the implication would be vacuously valid, so suppose it is valid with respect to Υ. Then there must be derivations of

1. $\vdash_\Sigma G[\![\theta]\!][\sigma]$ `ctx`,
2. $G[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and
3. $G[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$

in LF which satisfy the derivation height restriction identified by Ann with respect to Υ. Given that n_1 cannot appear in A_2, it also cannot appear in $A_2[\![\theta]\!]$ as the support of θ cannot contain n_1. Thus by Theorem 2.10 there are also derivations for $\vdash_\Sigma G'[\![\theta]\!][\sigma]$ `ctx`, $G'[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`, and $G'[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$. Furthermore, these derivations are of the same height as (1), (2), and (3) respectively, and thus must also respect the annotation Ann with respect to Υ. Therefore the formula $\{G' \vdash M : A\}^{Ann}[\![\theta]\!][\sigma]$ will be valid with respect to Υ, as needed.

Case: *LF-inst*

Let Υ be an arbitrary height assignment and θ and σ identify an arbitrary closed instance of the conclusion sequent. Suppose that all the formulas in $\Omega[\![\theta]\!][\sigma]$ are valid with respect to Υ since if any were not this closed instance would be vacuously valid. Consider the goal formula $(\{G, G' \vdash M : A\} \supset \{G \vdash M_1 : A_1\} \supset \{G'' \vdash M' : A'\})[\![\theta]\!][\sigma]$. If either $\{G \vdash M_1 : A_1\}[\![\theta]\!][\sigma]$ or $\{G, G' \vdash M : A\}[\![\theta]\!][\sigma]$ are not valid with respect to Υ, then this implication would be vacuously valid, so suppose that both of these formulas are valid. Then there must be derivations in LF for

1. $\vdash_\Sigma (G, G')[\![\theta]\!][\sigma]$ `ctx`,
2. $\vdash_\Sigma G[\![\theta]\!][\sigma]$ `ctx`,
3. $(G, G')[\![\theta]\!][\sigma] \vdash_\Sigma A[\![\theta]\!]$ `type`,
4. $G[\![\theta]\!][\sigma] \vdash_\Sigma A_1[\![\theta]\!]$ `type`,
5. $(G, G')[\![\theta]\!][\sigma] \vdash_\Sigma M[\![\theta]\!] \Leftarrow A[\![\theta]\!]$, and

6. $G[\![\theta]\!][\sigma] \vdash_\Sigma M_1[\![\theta]\!] \Leftarrow A_1[\![\theta]\!]$.

By Theorems 2.11 and 2.3 there are derivations of $\vdash_\Sigma G''[\![\theta]\!][\sigma]$ $\texttt{ctx}$, $G''[\![\theta]\!][\sigma] \vdash_\Sigma A'[\![\theta]\!]$ $\texttt{type}$, and $G''[\![\theta]\!][\sigma] \vdash_\Sigma M'[\![\theta]\!] \Leftarrow A'[\![\theta]\!]$. Thus the formula $\{G'' \vdash M' : A'\}\,[\![\theta]\!][\sigma]$ will be valid with respect to Υ, as needed. $\Box$

Chapter 5

Adelfa: An Implementation of the Proof System

Now that we have described a proof system for constructing validity arguments, we would like to provide a tool for mechanically constructing these proofs. We have implemented a proof assistant based on the logic which allows for the construction of proofs via a collection of tactics which correspond to the different reasoning steps that are available in the proof system. The sequents of the proof system correspond to states in the prover. The system was built in OCaml and has been used in a collection of reasoning examples, which are covered in the next chapter.

In the first section we will introduce the Adelfa reasoning system, it's structure and how it is used to mechanize the construction of proofs in our logic. There are a few special considerations in implementing this system which make up the remaining sections in this chapter. The *atm-app-L* rule depends on a function *AllCases* which generates sequents using a covering set of arity type preserving substitutions, and we must provide some realization of this process in Adelfa. In the second section we show that an implementation of higher-order pattern unification [Mil91, NL05] can be used to identify these covering sets of solutions such that they satisfy the necessary restrictions. The final section will look at the form of formulas used in the proof assistant. There are certain proof rules which may be permuted out to the end of a derivation, allowing us to work with a focused formula syntax in reasoning.

5.1 An Overview of the System

The architecture of the Adelfa system is influenced significantly by that of the Abella proof assistant [Gac09b, BCG^{+}14]. Adelfa has two levels of execution: the top level interaction and the proof level interaction. The former allows for the definition of an LF signature, context schemas, and theorems. Proved theorems are stored at this level as formulas which can be used as lemmas in later proofs. LF expressions in Adelfa follow Twelf syntax except that there are no types in abstraction terms as our logic is based on Canonical LF. We use `{G |- M : A}` to represent atomic formulas, $\top$ and $\bot$ become `true` and `false`, the connectives $\supset$, $\wedge$, and $\vee$ become `=>`, `/\`, and `\/` respectively, and the quantifiers $\forall$, $\exists$, and Π are denoted by `forall`, `exists`, and `ctx`.

The proof level is entered when a theorem is proposed. In essence, the proof states in Adelfa correspond to sequents of the proof system and represent that there is an obligation to provide a proof for that sequent. Proofs are constructed bottom-up using a defined set of tactics which correspond to a sequence of valid applications of proof rules. Some proof rules contain multiple sequents in the premises, thus a proof state in Adelfa will also keep track of all the remaining obligations as a stack.

Proof states in Adelfa are displayed in the following form.

```
Vars: T:o, E:o
Nominals: n1:o, n:o
Contexts: Gamma:c[(n:tm, n1:of n T)]
H1:{Gamma |- n : tm}
H2:{T : ty}


==================================
exists D3, {Gamma |- D3 : of n T}
```

Above the line are the components which are to the left of the sequent arrow in a sequent, and below the line is the goal formula which would appear on the right. Any further obligations in the stack are included below this state and are identified by the goal formula. We can see each of the sequent components identified here with both nominal constants and variables identified along with their arity types, and context variables identified along with their context variable type. The formulas in the assumption set appear with identifiers which are used to reference them in the system.

Central to the system are the tactics which are used to construct proofs of the theorems. Tactics are designed to capture only valid reasoning steps in the logic, and are also meant to capture the natural reasoning steps of a development. Figure 5.1 lists the tactics and provides a brief explanation of the result of applying the tactic in an Adelfa development.

The tactics `assert`, `exists`, `left`, `right`, and `split` encode the use of a single particular proof rule in reasoning, namely *cut*, $\exists$-R, $\vee$-R_i, and $\wedge$-R respectively. The `case` tactic uses the structure of a hypothesis to determine an applicable left rule from *atm-app-L*, $\wedge$-L_i, or $\vee$-L. Induction is realized through the `induction` tactic which captures an application of *ind*, which introduce the inductive hypothesis for reasoning inductively on the height of the identified typing judgement into the assumption formulas.

The tactics `apply`, `intros`, and `search` all involve following a procedure for applying multiple proofs rules in the given state. The application of hypotheses or lemmas is encoded by `apply`. Based on the structure of the formula being applied, this tactic encodes using proof rules $\forall$-L, Π-L, and $\supset$-L to instantiate the formula with the given arguments. Previously proved theorems are given by their identifier and would encode first using *cut* to introduce the formula and then following the above process. The `intros` tactic captures the introduction of variables and hypotheses for the outer quantifiers and implications. This encodes the application of some number of instances of the $\forall$-R, Π-R, and $\supset$-R proof rules based on the structure of the current goal formula. The `search` tactic attempts to automatically construct a proof for the current subgoal using *id*, *atm-app-R*, and *atm-abs-R*. The process is one which will always eventually find such a derivation if one exists given that the application of *atm-app-R* and *atm-abs-R* will decrease the size of the term in an atomic formula and exactly one of the two applies must apply to any closed term.

The remaining tactics, `weaken`, `strengthen`, `permutectx`, and `inst`, capture the application of an instance of an LF meta-theorems on the left. The general structure is one which uses *cut* to introduce the formula capturing the meta-theorem to be applied using the

Tactic	**Effect**
`apply H to H1 ... Hn with (G1 = g1), ..., n1 = t1, ...`	Extends the hypotheses with the result of applying lemma or hypothesis `H` with the given arguments.
`assert f`	Creates two new subgoals, one with `f` as the goal and another proving the current goal with `f` as an assumption.
`case H`	Applies an appropriate left to the hypothesis `H`.
`exists t`	Instantiates the outermost existential quantifier with `t`.
`induction on i`	Adds height annotations to the `i`th implication antecedent and introduces the inductive hypothesis to assumptions.
`inst H with n = t`	Instantiates the name `n` with `t` in the atomic hypothesis `H`.
`intros`	Introduces variables for outer universal and context quantifiers and hypotheses for outer implications.
`left`, `right`	Replaces goal `F1 \/ F2` with `F1` (`left`) or `F2` (`right`).
`permutectx H to G`	Permutes the context expression of atomic `H` to `G`.
`search`	Attempts to conclude the current subgoal for the user automatically.
`split`	Splits a goal `F1 /\ F2` into two subgoals for `F1` and `F2`.
`strengthen H`	Strengthens the context of atomic `H`.
`weaken H with t`	Weakens the context of atomic `H` with the type `t`.

Figure 5.1: Tactics of Adelfa

corresponding proof rule to determine any conditions that remain to be shown. Then the meta-theorem formula is applied using $\supset$-L to the appropriate hypotheses. The application of weakening also must first introduce a new name for the binding using *ctx-str*. In Adelfa these tactics are only applied successfully when the premises of the proof rule encoding the meta-theorem can be verified automatically.

We will demonstrate the use of these tactics in practice through the use of example Adelfa developments in Chapter 6.

5.2 Finding Covering Sets of Solutions for Case Analysis

The *atm-app-L* rule in the proof system provides a means for reasoning from atomic assumption formulas by analysing the possible reasons for their validity. The function *AllCases* is the main process defining the way in which cases are determined for a given formula. A key part of its definition is the notion of covering set of solutions to unification problems. The substitutions comprising this set must satisfy particular requirements that are identified in Definitions 4.16 and 4.20 respectively. For the discussion in this section, we assume that the set of equations $\mathcal{E}$ of a well-formed unification problem $\mathcal{U} = \langle \mathbb{N}; \Psi; \mathcal{E} \rangle$ comprise only equations between canonical terms and not canonical types. The reduction operation on a sequent also requires us to consider equations between atomic types. However these equations are of the form $a\ M_1 \ldots M_n = a\ M'_1 \ldots M'_n$ for some type level constant a in the signature or the two types in the equation have two different type constants as their heads. In the former case, the equation can be replaced by the set of equations $\{M_1 = M'_1, \ldots, M_n = M'_n\}$ and in the latter case the unification problem is seen immediately to have no solutions.

In light of the above observation, to implement the analysis embedded in the *AllCases* rule, we have to describe a procedure for finding a covering set of solutions to a set of equations between (arity-typed) λ-terms. This task is identified with solving a higher-order pattern unification problem [Mil91] and is addressed in Adelfa by using the higher-order pattern unification algorithm described by Nadathur and Linnell [NL05]. At a high level this procedure determines unifiers for terms by descending through their structure, ensuring that the fixed, non-variable parts are identical, eventually simplifying the problem to be solved to equations of the form $(x\ t_1\ \cdots\ t_n) = s$, where x is a term variable for which a substitution is to be considered. A characteristic of the higher-order pattern unification problem is that the terms $t_1, \ldots, t_n$ must all be nominal constants or variables bound by abstractions within whose scope the term $(x\ t_1\ \cdots\ t_n)$ and, correspondingly, s occurs. If this equation is solvable, then the substitution that is generated has the form of the term s enclosed within a sequence of abstractions binding the occurrences of $t_1, \ldots, t_n$ in s. The circumstances under which the equation is deemed solvable ensure that the generated substitution term will not contain any nominal constants. As a consequence, the support set for substitutions found by the higher-order pattern unification procedure will be disjoint from any set $\mathbb{N}$ of nominal constants. Moreover, the substitution terms will be typeable with respect to the arity type assignments under which the terms in the unification problem are well-typed augmented with suitable type assignments for any new term variables that are introduced in them, and the overall substitution will itself be arity type preserving with respect to these assignments. In short, the requirements 1-3 in Definition 4.16 will be satisfied by the solutions found by the higher-order pattern unification procedure and, being most-general unifiers, these solutions will also constitute (singleton) sets of covering solutions.

There is, however, one wrinkle to the use of higher-order pattern unification in Adelfa. This form of unification is defined for a more general form of λ-terms that includes non-canonical terms and in a setting in which substitution application does not include the concurrent reduction of terms to canonical form. We must therefore verify that the solutions it finds to a unification problem in which the terms are in canonical form also constitutes a solution in the sense described in this thesis. One requirement, that the substitution terms be in canonical form, is easily seen to be met: if the input terms are in canonical form, then the procedure we have outlined above will generate substitutions in which all the terms are in canonical form under the relevant arity typing. Thus it only remains to be seen that if two terms are determined to be equal under a substitution applied in the sense of the

$$\textbf{Terms} \quad T \quad ::= \quad c \mid x \mid \lambda x. T \mid T_1\, T_2$$

Figure 5.2: Extended Term Syntax

higher-order pattern unification procedure, then they are also equal when the substitution is applied in the sense relevant to our logic.

The remainder of this section will focus on arguing that this last requirement is also met. To do this we essentially argue that the reduction steps which are built into hereditary substitution can be separated out from the application of the substitution while ensuring that the resulting terms will be identical. To provide a framework for such an argument, we will introduce an extended term syntax and a notion of substitution application which does not reduce the result to a canonical form. We will then identify equivalence classes for terms in this extended syntax based on the λ-conversion rules under which terms are considered to be equal in the context of the higher-order pattern unification algorithm. The key result we will show is that the application of a hereditary substitution to a term in canonical form will produce a term that is in the same equivalence class as the term that is obtained by applying the substitution in the sense of the higher-order pattern unification algorithm to the term.

The extended term syntax is given in Figure 5.2 and it follows the standard syntax for λ-terms which allows for terms that contain β-redexes. Two terms are considered to be equal if one can be converted to the other using the usual rules of λ-conversion. We write $t_1 \equiv t_2$ to denote the fact that t_1 and t_2 are equal in the sense described. The terms that we consider here are ones that that are well-typed in an arity typing sense. In this context, every term will have a canonical form, i.e. a form in which there are no β-redexes and the top-level applications carry the arity type o. These canonical forms in fact constitute a subset of the the full set of terms in the extended syntax that is identical to the collection of well-formed terms in canonical LF. We will also view the canonical forms as distinguished representatives of the equivalence classes they belong to under the operative equality notion.

A substitution in this setting can be captured conveniently by creating a sequence of β-redexes; the resulting term will be in the same equivalence class as the term that results from an actual replacement, paying attention to renaming of bound variables to avoid accidental capture.

Definition 5.1. A substitution θ is a mapping from variables to terms in the extended term syntax written $\{t_1/x_1, \ldots, t_n/x_n\}$. The application of substitution $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ to a term T is written as $T[\theta]$ and is defined to be the term $\lambda x_1. \ldots \lambda x_n. (T\ t_1 \ldots t_n)$.

We now show that the result of hereditary substitution will be a term which, under the extended syntax for terms, is a member of the same equivalence class as a term obtained by applying the substitution following the simpler notion of application given in Definition 5.1. Clearly then, normalizing the term obtained using the simpler notion of substitution application will result in the same term, up to renaming, as that obtained using hereditary substitution.

Theorem 5.1. *Suppose that θ is an arity type preserving substitution with respect to Θ and $ctx(\theta) \uplus \Theta \vdash_{at} t : \alpha$ has a derivation. Letting $\theta' = \{t/x | \langle x, t, \alpha \rangle \in \theta\}$, $t[\theta'] \equiv t[\![\theta]\!]$.*

Proof. This proof is by induction using a lexicographic ordering of the size of θ, identified by the sum of the size of each type indexing θ, and the formation of the term t. We will consider each case based on the structure of t.

Case: $\mathbf{t} = \mathbf{x}$ for some $\mathbf{x} : \alpha \in ctx(\theta) \uplus \mathbf{\Theta}$

If $x \in dom\,(\theta)$ then $x[\theta'] = t_x$ for $\langle x, t_x, \alpha \rangle \in \theta$ and therefore $x[\theta'] \equiv x[\![\theta]\!]$ will clearly hold. If $x \notin dom\,(\theta)$ then $x[\theta'] = x$, and $x[\theta'] \equiv x[\![\theta]\!]$ in this case as well.

Case: $\mathbf{t} = (\mathbf{t_1}\ \mathbf{t_2})$

Then there will be derivations for the arity typing judgements $ctx(\theta) \uplus \Theta \vdash_{at} t_1 : \alpha' \to \alpha$ and $ctx(\theta) \uplus \Theta \vdash_{at} t_2 : \alpha'$. By induction we can determine that $t_1[\theta'] \equiv t_1[\![\theta]\!]$ and $t_2[\theta'] \equiv t_2[\![\theta]\!]$. It is not difficult to see that $t[\theta'] \equiv (t_1[\theta'])\ (t_2[\theta'])$, and thus that $t[\theta'] \equiv (t_1[\![\theta]\!])\ (t_2[\![\theta]\!])$. If $t_1[\![\theta]\!]$ is not an abstraction term then $t[\![\theta]\!] = (t_1[\![\theta]\!])\ (t_2[\![\theta]\!])$, and therefore $t[\theta'] \equiv t[\![\theta]\!]$ will clearly hold. If on the other hand $t_1[\![\theta]\!] = \lambda y. s : \alpha' \to \alpha$, then $t[\![\theta]\!] = s[\![\{\langle y, (t_2[\![\theta]\!]), \alpha' \rangle\}]\!]$.

In this case we can see that the equivalence $t[\theta'] \equiv (\lambda y.\, s)\ (t_2[\![\theta]\!])$ will hold. Using an application of β-reduction for the top-level β-redex in $(\lambda y.\, s)\ (t_2[\![\theta]\!])$, we see that $t[\theta']$ is then equivalent to $s[(t_2[\![\theta]\!])/y]$. From the derivation of $ctx(\theta) \uplus \Theta \vdash_{at} t_1 : \alpha' \to \alpha$ and that θ is arity type preserving with respect to Θ we can determine that $y : \alpha' \uplus \Theta \vdash_{at} s : \alpha$ has a derivation. The size of the type α' must be strictly smaller than the size of the types indexing θ by the definition of hereditary substitution, and thus we can apply the induction hypothesis to conclude that $s[(t_2[\![\theta]\!])/y] \equiv s[\![\{\langle y, (t_2[\![\theta]\!]), \alpha'\rangle\}]\!]$. Therefore, $t[\theta'] \equiv t[\![\theta]\!]$.

Case: $\mathbf{t} = \lambda \mathbf{x}.\, \mathbf{t}'$

Then $\alpha = \alpha_1 \to \alpha_2$ and there is a derivation of $ctx(\theta) \uplus \Theta, x : \alpha_1 \vdash_{at} t' : \alpha_2$. By induction then, $t'[\theta'] \equiv t'[\![\theta]\!]$. Since $t[\theta'] \equiv \lambda x.\, (t'[\theta'])$ and $t[\![\theta]\!] = \lambda x.\, (t'[\![\theta]\!])$, we can thus conclude that $t[\theta'] \equiv t[\![\theta]\!]$. □

For θ and $\theta' = \{t/x | \langle x, t, \alpha\rangle \in \theta\}$ satisfying the arity typing requirements of the above theorem, we see that for any canonical terms M and M' if $M[\theta'] = M'[\theta']$ we can apply the above lemma to conclude that $M[\![\theta]\!] = M'[\![\theta]\!]$ also holds. Therefore since higher-order pattern unification ensures that $M[\theta'] = M'[\theta']$ the substitutions found following this procedure will also be solutions in the sense of Definition 4.16 and therefore identify a covering set of solutions, as described at the beginning of this section.

5.3 Focusing Formulas

In this section we consider the construction of proofs in the logic and note that there are some rules which are permutable such that they can be moved to the end of a derivation and thus be applied automatically in the reasoning system. This leads to a focusing of formulas in the reasoning system to ones of a restricted structure. The rules that we will consider in this discussion are *atm-abs-L* and *∃-L*.

Given the LF typing rules we note that $\Gamma \vdash_\Sigma \lambda x.\, M \Leftarrow \Pi x{:}A_1.\, A_2$ is derivable if and only if $\Gamma, x : A_1 \vdash_\Sigma M \Leftarrow A_2$ is derivable. The following theorem lifts this idea to the derivability of sequents containing atomic assumption formulas involving abstraction terms. We argue that for any such sequent which is derivable there must also be a derivation for the sequent which concludes by an application of *atm-abs-L*. In this way, we see that it is sound to reduce all atomic assumption formulas of a sequent to be typing judgements over atomic terms.

Theorem 5.2. *Suppose the sequent* $\mathbb{N}; \Psi; \Xi; \Omega, \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\} \longrightarrow F$ *is derivable. Let* n *be a new nominal constant not appearing in* $\mathbb{N}$*,* M' *be* $M[\![\{\langle x, n, (A_1)^- \rangle\}]\!]$*,* A_2' *be* $A_2[\![\{\langle x, n, (A_1)^- \rangle\}]\!]$*, and* Ξ' *be* $(\Xi \setminus \{\Gamma_i \uparrow \mathbb{N}_i : \mathcal{C}_i[\mathcal{G}_i]\}) \cup \{\Gamma_i \uparrow (\mathbb{N}_i, n : (A_1)^-) : \mathcal{C}_i[\mathcal{G}_i]\}$ *if* G *contains a context variable* Γ_i *or* Ξ *if* G *contains no context variables. Then the sequent* $\mathbb{N}, n : (A_1)^-; \Psi; \Xi'; \Omega, \{G, n : A_1 \vdash M' : A_2'\} \longrightarrow F$ *must be derivable.*

Proof. Let $\mathcal{D}$ be the derivation for $\mathbb{N}; \Psi; \Xi; \Omega, \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\} \longrightarrow F$. Applying first *ctx-wk* followed by *weak* to the derivation $\mathcal{D}$ we construct a derivation for the sequent $\mathbb{N}, n : (A_1)^-; \Psi; \Xi'; \Omega, \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}, \{G, n : A_1 \vdash M' : A_2'\} \longrightarrow F$. Using an application of *id* followed by *atm-abs-R* we can also construct a derivation for the sequent $\mathbb{N}, n : (A_1)^-; \Psi; \Xi'; \Omega, \{G, n : A_1 \vdash M' : A_2'\} \longrightarrow \{G \vdash \lambda x.\, M : \Pi x{:}A_1.\, A_2\}$. Finally, an application of *cut* using these two derivations will construct a derivation for the sequent $\mathbb{N}, n : (A_1)^-; \Psi; \Xi'; \Omega, \{G, n : A_1 \vdash M' : A_2'\} \longrightarrow F$ as needed. □

The same basic structure is used to argue for the reduction of existential formulas in assumptions using $\exists$-*L*. The essence of the argument is that we can recover the validity of the existential formula, and thus do not lose any derivations by applying $\exists$-*L* eagerly in Adelfa.

Theorem 5.3. *Let* $\mathbb{N} = n_1 : \alpha_1, \ldots, n_m : \alpha_m$ *be a collection of arity typed nominal constants. Suppose that the sequent* $\mathbb{N}; \Psi; \Xi; \Omega, \exists x : \alpha.F_1 \longrightarrow F_2$ *has a derivation. Let* y *be an arbitrary new variable of arity type* $\alpha' = (\alpha_1 \to \ldots \to \alpha_m \to \alpha)$*,* $\Psi' = \Psi, y : \alpha'$*, and* $F_1[\![\{\langle x, (y\ n_1 \ldots n_m), \alpha \rangle\}]\!] = F_1'$*. Then the sequent* $\mathbb{N}; \Psi'; \Xi; \Omega, F_1' \longrightarrow F_2$ *has a derivation.*

Proof. Let $\mathcal{D}$ be the derivation for $\mathbb{N}; \Psi; \Xi; \Omega, \exists x : \alpha.F_1 \longrightarrow F_2$. From $\mathcal{D}$ using an application of *ctx-wk* followed by *weak* we construct a derivation for $\mathbb{N}; \Psi'; \Xi; \Omega, \exists x : \alpha.F_1, F_1' \longrightarrow F$. We also construct a derivation for the sequent $\mathbb{N}; \Psi'; \Xi; \Omega, F_1' \longrightarrow \exists x : \alpha.F_1$ from an application of *id* followed by $\exists$-R. From these two derivations we use an application of *cut* to construct a derivation for the sequent $\mathbb{N}; \Psi'; \Xi; \Omega, F_1' \longrightarrow F$ as needed. $\square$

Given these results, the system Adelfa will automatically reduce assumption formulas to have no top-level existential quantifiers and ensure that any atomic formulas in the assumptions are in the form of an atomic LF term.

Chapter 6

Constructing Proofs Using Adelfa

In this chapter we demonstrate the construction of proofs in Adelfa using tactics. Our first example is a proof of the existence of an additive identity for natural numbers. This example is used to introduce the basic structure of reasoning in Adelfa and demonstrates the expressiveness of the logic as the statement of the theorem contains two quantifier alternations. We then consider the example of type uniqueness, which we use to introduce reasoning by induction, and the use of case analysis, and the role of contexts in Adelfa reasoning. Next we present an encoding for a simple sequent calculus and consider proving that cut is admissible in this system. Through this example we will bring out how the meta-theorems of LF are used in reasoning using Adelfa. We conclude this chapter with a discussion of the transitivity and narrowing of $F_{<:}$, Problem 1A of the POPLMark Challenge [ABF^{+}05]. The complete development for all of these examples can be found on the Adelfa website.

$$
\begin{array}{ll}
\mathit{nat} : \mathrm{Type} & \mathit{plus} : \mathit{nat} \to \mathit{nat} \to \mathit{nat} \to \mathrm{Type} \\
z : \mathit{nat} & \mathit{plus\_z} : \Pi N{:}\mathit{nat}.\, \mathit{plus}\ z\ N\ N \\
s : \mathit{nat} \to \mathit{nat} & \mathit{plus\_s} : \Pi N_1{:}\mathit{nat}.\, \Pi N_2{:}\mathit{nat}.\, \Pi N_3{:}\mathit{nat}. \\
 & \qquad\qquad \Pi D{:}\mathit{plus}\ N_1\ N_2\ N_3.\, \mathit{plus}\ (s\ N_1)\ N_2\ (s\ N_3)
\end{array}
$$

Figure 6.1: An LF Specification for Natural Number Addition

6.1 Additive Identity for Natural Numbers

An encoding of natural numbers and addition over these expressions is given in Figure 6.1. There is a single type *nat* for representing natural numbers in the system and the type family *plus* represents the addition relation over these terms. This signature will be presented to Adelfa as the relevant LF signature to be used in reasoning.

The existence of an identity for the addition relation is captured by the following formula.

$$\exists i : o. \forall x : o. \{\cdot \vdash x : nat\} \supset \exists d : o. \{\cdot \vdash d : plus\ x\ i\ x\}$$

In particular, this states that there is a right identity for the relation. Informally, the proof must first introduce the identity term, z, and argue that for this instance the formula will be valid. Since the encoding of *plus* we have given is recursive in the first argument, the proof of this theorem will be by induction on the formation of x which identifies two structures, $x = z$ or $x = s\,x'$. For the former case, ($plus\_z\ z$) clearly inhabits the required type while for the latter case we make use of the inductive hypothesis on the smaller term x' and construct an appropriate term from this result using *plus_s*. The development in Adelfa, as we will see below, follows this structure.

The Adelfa development for this theorem begins by introducing the identity term through an application of the `exists` tactic with the expression z. This will result in the following state.

```
Vars:
Nominals:
Contexts:
==================================
forall x:o, {x : nat} => exists d:o, {d : plus x z x}
```

We now want to construct an inductive argument on the formation of x, which is captured in Adelfa through the use of `induction` with the argument 1 to identify that the first antecedent is the LF derivation on which the induction is to be based. The result of the application of this tactic is the following state.

```
Vars:
Nominals:
Contexts:
IH: forall x:o, {x : nat}* => exists d:o, {d : plus x z x}
==================================
forall x:o, {x : nat}@ => exists d:o, {d : plus x z x}
```

At this stage we introduce a new eigenvariable for x, and add the formula `{x : nat}@` to the assumptions through an application of the `intros` tactic. Following the use of this tactic the new state of the development will be the following.

```
Vars: x:o
Nominals:
Contexts:
IH: forall x:o, {x : nat}* => exists d:o, {d : plus x z x}
H1: {x : nat}@
==================================
exists d:o, {d : plus x z x}
```

An application of `case` on `H1` will identify two structures for the derivation of $\{\vdash x : nat\}$ and thus two subgoals in Adelfa, corresponding to the base and inductive case of the informal argument. The subgoal corresponding to the case where $x = z$ is the following.

```
Vars: x:o
Nominals:
Contexts:
IH: forall x:o, {x : nat}* => exists d:o, {d : plus x z x}
H1: {z : nat}@
==================================
exists d:o, {d : plus z z z}
```

We can conclude this subgoal through instantiating the existential with an appropriate term through the tactic `exists (plus_z z)`, and invoking `search` to have Adelfa automatically determine that the goal is derivable using the proof rules for deriving atomic formulas. Once this subgoal is completed, we must then argue for the inductive case, represented in the following state.

```
Vars: x:o, x':o
Nominals:
Contexts:
IH: forall x:o, {x : nat}* => exists d:o, {d : plus x z x}
H1: {s x' : nat}@
H2: {x' : nat}*
==================================
exists d:o, {d : plus (s x') z (s x')}
```

We now make use of the inductive hypothesis through the application of the `apply` tactic, `apply IH to H2`, which will introduce an instance of the conclusion derivation for some new eigenvariable d.

```
Vars: x:o, x':o, d:o
Nominals:
Contexts:
IH: forall x:o, {x : nat}* => exists d:o, {d : plus x z x}
H1: {s x' : nat}@
H2: {x' : nat}*
H3: {d : plus x' z x'}
==================================
exists d:o, {d : plus (s x') z (s x')}
```

We complete this proof using this new eigenvariable to instantiate the d in the goal formula with a term $plus\_s\ x'\ z\ d$ using `exists`. At this point the `search` tactic can be used and

$$
\begin{array}{ll}
tp : \mathrm{Type} & \mathit{of\_empty} : \mathit{of}\ \mathit{empty}\ \mathit{unit} \\
\mathit{unit} : tp & \\
\mathit{arr} : tp \to tp & \mathit{of\_app} : \Pi E_1{:}tm.\, \Pi E_2{:}tm.\, \Pi T_1{:}tp.\, \Pi T_2{:}tp. \\
 & \quad \Pi D_1{:}\mathit{of}\ E_1\ (\mathit{arr}\ T_1\ T_2).\, \Pi D_2{:}\mathit{of}\ E_2\ T_1. \\
tm : \mathrm{Type} & \quad \mathit{of}\ (\mathit{app}\ E_1\ E_2)\ T_2 \\
\mathit{empty} : tm & \\
\mathit{app} : tm \to tm \to tm & \mathit{of\_lam} : \Pi R{:}tm \to tm.\, \Pi T_1{:}tp.\, \Pi T_2{:}tp. \\
\mathit{lam} : tp \to (tm \to tm) \to tm & \quad \Pi D{:}(\Pi x{:}tm.\, \Pi y{:}\mathit{of}\ x\ T_1.\, \mathit{of}\ (R\ x)\ T_2). \\
 & \quad \mathit{of}\ (\mathit{lam}\ T_1\ (\lambda x.\, R\ x))\ (\mathit{arr}\ T_1\ T_2) \\
\mathit{of} : tm \to tp \to \mathrm{Type} & \\
\mathit{eq} : tp \to tp \to \mathrm{Type} & \mathit{refl} : \Pi T{:}tp.\, \mathit{eq}\ T\ T
\end{array}
$$

Figure 6.2: An LF Specification for the Simply-Typed Lambda Calculus

Adelfa will automatically determine the goal is derivable. At this point there are no further subgoals representing further proof obligations, and the derivation is completed.

6.2 Type Uniqueness for the STLC

In this section we consider the example of type uniqueness for the STLC to see how Adelfa is used to reason about systems involving binding. Proving this property involves reasoning inductively over typing judgements which requires generalizing over the contexts of these typing judgements. Using the same signature presented in Section 2.2, and the discussion of the argument structure from Section 3.3 we look at how this derivation is formalized in Adelfa. To make it easy to follow the discussion of the reasoning process, we have reproduced the specification in Figure 6.2.

As in the informal discussion, an Adelfa development of this proof will introduce a context schema defined by a single block, $\{t : o\}x : tm, y : \mathit{of}\ x\ t$. We name this schema as *ctx* and use this name to refer to it in the discussion that follows.

In this proof we rely on two strengthening properties, one for terms of LF type tp, $\Pi\,\Gamma : ctx.\forall t : o.\,\{\Gamma \vdash t : tp\} \supset \{\cdot \vdash t : tp\}$, and building on this one for LF terms of any instance of the eq type, $\Pi\,\Gamma : ctx.\forall d : o.\forall t_1 : o.\forall t_2 : o.\,\{\Gamma \vdash d : eq\ t_1\ t_2\} \supset \{\cdot \vdash d : eq\ t_1\ t_2\}$. These are clearly valid formulas given the context schema ctx and the strengthening of Canonical LF judgements, Theorem 2.9, but they are also derivable formulas in the proof system by an induction on the assumption formula. Case analysis on the assumption will yield only terms constructed from constants in the signature, and thus nothing from the context can appear in these judgements.

The formula representing type uniqueness which we prove in Adelfa is the following.

$$\Pi\,G : ctx.\forall e : o.\forall t_1 : o.\forall t_2 : o.\forall d_1 : o.\forall d_2 : o.\,\{G \vdash d_1 : \mathit{of}\ e\ t_1\} \supset$$
$$\{G \vdash d_2 : \mathit{of}\ e\ t_2\} \supset \exists d_3 : o.\,\{\cdot \vdash d_3 : eq\ t_1\ t_2\}$$

The informal argument is based on an induction on the derivation of $\{G \vdash d_1 : \mathit{of}\ e\ t_1\}$, and in Adelfa this is accomplished using the `induction` tactic with argument 1 to indicate that this first formula is the one on which we wish to induct. The resulting state, now containing annotated formulas, is of the following form.

```
Vars:
Nominals:
Contexts:
IH: ctx G:ctx, forall e:o t1:o t2:o d1:o d2:o, {G |- d1 : of e t1}* =>
      {G |- d2 : of e t2} => exists d3:o, {d3 : eq t1 t2}
==================================
ctx G:ctx, forall e:o t1:o t2:o d1:o d2:o, {G |- d1 : of e t1}@ =>
  {G |- d2 : of e t2} => exists d3:o, {d3 : eq t1 t2}
```

After introducing the assumptions using `intros` the proof will proceed by determining the possible derivations for instances of $\{G \vdash d_1 : \mathit{of}\ e\ t_1\}^{@}$. We use the `case` tactic on this assumption formula, having Adelfa analyse the given assumption formula and replace this goal with a set of new subgoals. Adelfa will identify the same four cases discussed in the informal argument: when the head of d_1 is *of_empty*, *of_app*, *of_lam*, or a nominal constant assigned the type $(of\ n\ t_1)$ in G. In all of these cases, the form of d_1 constrains the form of e which in turn will constrain the derivable instances for the other assumption formula $\{G \vdash d_2 : of\ e\ t_2\}$ to those where the head of d_2 is the same as that of d_1. Thus in each of the four cases we identify this constrained structure by using the `case` tactic also on this second assumption formula to unfold the definition of this term-level constant.

When the head of d_1 is *of_empty* the proof will conclude using an application of the `exists` tactic with the obvious *refl* term followed by `search` to complete the derivation. In the cases where the head of d_1 is *of_app* or *of_lam* we invoke the induction hypothesis using the `apply` tactic with the appropriate hypotheses as arguments. To demonstrate, let us consider the case for typing abstraction terms.

```
Vars: a2:o -> o -> o, t2:o, a1:o -> o -> o, r:o -> o, t:o, t1:o
Nominals: n1:o, n:o
Contexts: G:ctx[]
IH:ctx G:ctx, forall e:o t1:o t2:o d1:o d2:o, {G |- d1 : of e t1}* =>
      {G |- d2 : of e t2} => exists d3:o, {d3 : eq t1 t2}
H3:{G, n:tm |- r n : tm}*
H4:{G |- t : ty}*
H5:{G |- t1 : ty}*
H6:{G, n:tm, n1:of n t |- a1 n n1 : of (r n) t1}*
H10:{G |- t : ty}
H11:{G |- t2 : ty}
H12:{G, n:tm, n1:of n t |- a2 n n1 : of (r n) t2}
==================================
exists d3:o, {d3 : eq (arr t t1) (arr t t2)}
```

Adelfa is able to identify that the extended context expression $(G,\ n : tm,\ n1 : of\ n\ t)$ satisfies the context schema *ctx* given that G is a context variable satisfying this same schema. Thus we are able to apply the formula `IH` to `H6` and `H12` to conclude that there is some $d3$ such that $\{\cdot \vdash d3 : eq\ t1\ t2\}$ is a valid formula. We know from the definition of the type *eq* that this means $t1 = t2$ and $d3$ is the term (*refl* $t1$), and using `case` in Adelfa will add this information resulting in the following state.

```
Vars: a2:o -> o -> o, a1:o -> o -> o, r:o -> o, t:o, t1:o
Nominals: n1:o, n:o
Contexts: G:ctx[]
IH:ctx G:ctx, forall e:o t1:o t2:o d1:o d2:o, {G |- d1 : of e t1}* =>
      {G |- d2 : of e t2} => exists d3:o, {d3 : eq t1 t2}
H3:{G, n:tm |- r n : tm}*
H4:{G |- t : ty}*
H5:{G |- t1 : ty}*
H6:{G, n:tm, n1:of n t |- a1 n n1 : of (r n) t1}*
H10:{G |- t : ty}
H11:{G |- t1 : ty}
H12:{G, n:tm, n1:of n t |- a2 n n1 : of (r n) t1}
H13:{refl t1 |- eq t1 t1}
==================================
exists d3:o, {d3 : eq (arr t t1) (arr t t1)}
```

One may expect that after an application of `exists` with the term (*refl* (*arr* t $t1$)) this case is completed by an application of `search`. However, this does not succeed as we still need to use the strengthening properties to ensure that (*arr* t $t1$) is a good type expression in an empty context. The `apply` tactic is used for this, to apply the previously proved theorem as a lemma.

In the final case, where the head of $d1$ is a nominal constant assigned the type (*of* n $t1$) in G the context type for G is extended to include such a block, and $d2$ must also be this same nominal constant given that there can be at most one block containing an assignment for the nominal constant n. This case is concluded through an application of the strengthening lemma to ensure that the type $t1$ does not rely on anything from G.

$$\frac{}{\Gamma, C \Rightarrow C}\ \text{init} \qquad \frac{}{\Gamma \Rightarrow \top}\ \top R \qquad \frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\ \wedge R$$

$$\frac{\Gamma, A \wedge B, A, B \Rightarrow C}{\Gamma, A \wedge B \Rightarrow C}\ \wedge L \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ \supset R \qquad \frac{\Gamma, A \supset B \Rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\ \supset L$$

Figure 6.3: A Simple Intuitionistic Sequent Calculus

Once we have a derivation for the formula

$$\Pi\, G : ctx.\forall e : o.\forall t_1 : o.\forall t_2 : o.\forall d_1 : o.\forall d_2 : o. \{G \vdash d_1 : of\ e\ t_1\} \supset$$
$$\{G \vdash d_2 : of\ e\ t_2\} \supset \exists d_3 : o. \{\cdot \vdash d_3 : eq\ t_1\ t_2\}$$

in our development, the special case

$$\forall e : o.\forall t_1 : o.\forall t_2 : o.\forall d_1 : o.\forall d_2 : o. \{\cdot \vdash d_1 : of\ e\ t_1\} \supset$$
$$\{\cdot \vdash d_2 : of\ e\ t_2\} \supset \exists d_3 : o. \{\cdot \vdash d_3 : eq\ t_1\ t_2\}$$

is derivable directly from an application to the empty context.

6.3 Admissibility of Cut for a Simple Sequent Calculus

This example deals with the encoding of a very simple sequent calculus which is defined through the rules given in Figure 6.3. A sequent comprises a collection of hypotheses Γ and a conclusion represented by a proposition C. Our goal with this example is to prove the admissibility of cut for this object system. This property would is captured by the following: for any A, if $\Gamma \Rightarrow A$ and $\Gamma, A \Rightarrow C$ then $\Gamma \Rightarrow C$. The informal structure of this proof, upon which the Adelfa formalization will be based, is a standard one; we argue that the application of cut with A can be permuted up the proof of C and uses of the cut formula A are eliminated in the derivation of C essentially by replacing uses of the hypothesis with the proof for A. This argument is structured as a nested induction on the structure of the cut formula A and the structure of the derivation for C using the cut formula. The proof then proceeds by examining the structure of the derivation of C using the cut formula.

For cases which do not make use of the cut formula A in the final rule application of the derivation we simply apply the inductive hypothesis to each branch of the derivation and use the same rule application to construct a cut-free proof using the cut-free proofs for each branch. When the cut formula is relevant to the final step of the derivation, which is those cases where the derivation concludes by using one of the left rules on the cut formula A, we make use of the induction on the cut formula to replace the use of the hypothesis with an argument structured around the formation of A which essentially re-create the proof of A in place to produce a cut-free proof. In this proof, management of the hypotheses Γ will play an important role. In particular, as we descend through the structure of the proof for C the hypotheses may be extended and the proof for A must correspondingly be extended for the statement to apply inductively. Thus this proof relies on weakening in the sequent calculus which must be realized also in our formalization of the proof.

The LF specification for the sequent calculus is given in Figure 6.4. Key to this encoding is the way in which sequents are represented. In the encoding we make use of two LF types, *hyp* and *conc*, to identify propositions which are hypotheses and conclusions of a sequent respectively, thus the sequent $A_1, \ldots A_n \Rightarrow B$ will be represented by a judgement $x_1 : \mathit{hyp}\ A_1, \ldots, x_n : \mathit{hyp}\ A_n \vdash_\Sigma c \Leftarrow \mathit{conc}\ B$. The contexts we are interested in for this example will thus contain bindings only for instances of the *hyp* type. We define a context schema for such LF contexts of the form $\{a : o\}x : (\mathit{hyp}\ a)$ and associate this schema with the identifier c. Using this schema definition, cut admissibility can be stated as the following formula.

$$\Pi\, G : c.\forall a : o.\forall b : o.\forall d_1 : o.\forall d_2 : o \to o.$$
$$\{\cdot \vdash a : \mathit{prop}\} \supset \{G \vdash d_1 : \mathit{conc}\ a\} \supset \{G, n : \mathit{hyp}\ a \vdash d_2\ n : \mathit{conc}\ b\} \supset$$
$$\exists d : o. \{G \vdash d : \mathit{conc}\ b\}\,.$$

Note that in the third antecedent the term variable d_2 is permitted to depend on the cut formula a through the explicit dependency on the nominal constant n bound in the context.

The proof for this formula uses a nested induction first on the structure of a, through

$$
\begin{aligned}
&\mathit{prop} : \mathrm{Type}\\
&\mathit{top} : \mathit{prop}\\
&\mathit{imp} : \mathit{prop} \to \mathit{prop} \to \mathit{prop}\\
&\mathit{and} : \mathit{prop} \to \mathit{prop} \to \mathit{prop}\\
\\
&\mathit{hyp} : \mathit{prop} \to \mathrm{Type}\\
&\mathit{conc} : \mathit{prop} \to \mathrm{Type}\\
\\
&\mathit{init} : \Pi A{:}\mathit{prop}.\, \Pi D{:}\mathit{hyp}\ A.\, \mathit{conc}\ A\\
&\mathit{topR} : \mathit{conc}\ \mathit{top}\\
&\mathit{andL} : \Pi A{:}\mathit{prop}.\, \Pi B{:}\mathit{prop}.\, \Pi C{:}\mathit{prop}.\, \Pi D1{:}(\Pi x{:}\mathit{hyp}\ A.\, \Pi y{:}\mathit{hyp}\ B.\, \mathit{conc}\ C).\\
&\qquad \Pi D2{:}\mathit{hyp}\ (\mathit{and}\ A\ B).\, \mathit{conc}\ C\\
&\mathit{andR} : \Pi A{:}\mathit{prop}.\, \Pi B{:}\mathit{prop}.\, \Pi D_1{:}\mathit{conc}\ A.\, \Pi D_2{:}\mathit{conc}\ B.\, \mathit{conc}\ (\mathit{and}\ A\ B)\\
&\mathit{impL} : \Pi A{:}\mathit{prop}.\, \Pi B{:}\mathit{prop}.\, \Pi C{:}\mathit{prop}.\, \Pi D_1{:}\mathit{conc}\ A.\, \Pi D_2{:}(\Pi x{:}\mathit{hyp}\ B.\, \mathit{conc}\ C).\\
&\qquad \Pi D_3{:}\mathit{hyp}\ (\mathit{imp}\ A\ B).\, \mathit{conc}\ C\\
&\mathit{impR} : \Pi A{:}\mathit{prop}.\, \Pi B{:}\mathit{prop}.\, \Pi D{:}(\Pi x{:}\mathit{hyp}\ A.\, \mathit{conc}\ B).\, \mathit{conc}\ (\mathit{imp}\ A\ B)
\end{aligned}
$$

Figure 6.4: An LF Specification for Derivations

the first assumption formula $\{\cdot \vdash a : \mathit{prop}\}$, and second on the height of the derivation for b which may use a, through the second assumption formula $\{G, n : \mathit{hyp}\ a \vdash d_2\ n : \mathit{conc}\ b\}$. One thing to note about this proof is that given the meaning of LF judgements, the context expression $(G, n : \mathit{hyp}\ a)$ in this formula requires that the assumption of the cut formula a is always at the end of the context for this property to apply. A property of the object system is that any hypothesis from Γ can be identified for use in the application of the rules in Figure 6.3. The encoding for sequents we have defined is such that this property of the object system is realized by the permutation meta-theorem for LF; given the signature defined in Figure 6.4 we can determine that there can be no valid dependencies between types in contexts satisfying the schema c. Therefore any permutation will be valid, and in particular the permutation which moves the assumption for a to the end will be valid. Our encoding of the object system also permits weakening in the object system to be realized by weakening in LF. We demonstrate how these meta-theorems are applied in reasoning to manage the structure of contexts by looking at the formalization of the cut elimination proof in Adelfa. In the discussion below we look in detail at a single case of the argument to demonstrate how these meta-theorems can be used to realize the informal proof structure. The full details of this proof can be found on the Adelfa website, and with an understanding of the informal structure it is not difficult to see how the rest of the argument will be formalized.

The case we consider is when the derivation making use of the cut formula has a conclusion which is an implication, and this derivation concludes with a use of $\supset R$; this of course corresponds to the case where $(d_2\ n)$ is an instance of *impR* in Adelfa. This is a case where the cut formula is not used in the concluding rule of the derivation, so following the informal argument structure we use the inductive hypothesis to obtain cut-free proofs for the premises and then use these to construct a cut-free proof for the conclusion using an application of *impR*. The state of Adelfa at the start of this case has the following form.

```
Vars: d:o -> o -> o, b1:o, b2:o, d1:o, a:o
Nominals: n1:o, n:o
Contexts: G:c[]
IH:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}* => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b} => exists d, {G |- d : conc b}
IH1:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}@ => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b}** => exists d, {G |- d : conc b}
H1:{a : prop}@
H2:{G |- d1 : conc a}
H3:{G, n:hyp a |- impR b1 b2 ([x] d n x) : conc (imp b1 b2)}@@
H4:{G, n:hyp a |- b1 : prop}**
H5:{G, n:hyp a |- b2 : prop}**
H6:{G, n:hyp a, n1:hyp b1 |- d n n1 : conc b2}**
==================================
exists d, {G |- d : conc (imp b1 b2)}
```

Intuitively, we would like at this point to apply the inductive hypothesis `IH1` using `H1`, `H2`, and `H6` to extend the assumptions with a new cut-free derivation d' of the type $(\mathit{conc}\ b_2)$. However, to successfully apply the inductive hypothesis the assignment $n : (\mathit{hyp}\ a)$ must be moved to the end of the context expression. Further, the context expression in `H2` will need to be extended with an assignment to the type $(\mathit{hyp}\ b_1)$ to ensure that the context of this judgement matches the extended context in `H6`. To permute the cut formula to the end of the context expression, we will use the `permutectx` tactic on `H6` with the permuted context expression $(G, n_1 : \mathit{hyp}\ b_1, n : \mathit{hyp}\ a)$. This application will succeed since there can be no dependency on n in the type $(\mathit{hyp}\ b_1)$, which aligns with the interpretation of hypotheses in the object system. Adelfa uses the structure of the hypothesis `H6` and the permuted form of the context, $(G, n_1 : \mathit{hyp}\ b_1, n : \mathit{hyp}\ a)$, supplied by the user to identify a particular instance of the formula encoding the LF context permutation theorem. In this particular case that formula would be

$$\{G, n : hyp\ a, n_1 : hyp\ b_1 \vdash d\ n\ n_1 : conc\ b_2\}^{**} \supset \\ \{G, n_1 : hyp\ b_1, n : hyp\ a \vdash d\ n\ n_1 : conc\ b_2\}^{**}$$

which, as we have already noted, will be valid given that this permutation will not violate any dependencies. The effect of introducing such a formula using *cut* and then applying it to the hypothesis `H6` is the behaviour exhibited in Adelfa, and the resulting state includes a new assumption formula

```
H7:{G, n1:hyp b1, n:hyp a |- d n n1 :  conc b2}**.
```

The application of the `permutectx` tactic will fail if Adelfa is not able to determine that the order of bindings in the permuted context expression does not violate any dependencies, and so this tactic will only succeed when the use of this proof rule is valid.

We now want to realize the weakening step of the proof to extend the hypotheses of the proof for a with b_1. Since weakening in the object system is realized by weakening in LF, we achieve this through an application of the `weaken` tactic to the assumption formula `H2` and the type ($hyp\ b_1$). In this case the instance of the weakening formula identified by Adelfa will be

$$\{G \vdash d_1 : conc\ a\} \supset \{G, n_2 : hyp\ b_1 \vdash d_1 : conc\ a\}.$$

However, the application of this tactic to `H2` will fail because Adelfa is not able to ensure that all the premise sequents which would be generated by the use of *LF-wk* in proving this formula can be derived from the current state. Specifically, Adelfa is unable to automatically derive the subgoal $\{G \vdash b_1 : prop\}$. We do, however, have as assumption `H4` the formula $\{G, n : hyp\ a \vdash b_1 : prop\}^{**}$, and the formation of this b_1 cannot depend on anything of type *hyp*. So in particular, b_1 cannot depend on n, meaning this binding is vacuous. An application of the strengthening meta-theorem using the `strengthen` tactic will result in an assumption of the form necessary for the weakening step to succeed. Similar to the behaviour with context permutation and weakening, Adelfa introduces an instance of *LF-str* based on the hypothesis being strengthened and extends the assumption set with the result of applying this instance to the identified formula. In this case, the result would be to extend the state with a formula `H8:{G |- b1 :  prop}**`. At this point weakening can be successfully applied to `H2`, producing the following state.

```
Vars: d:o -> o -> o, b1:o, b2:o, d1:o, a:o
Nominals: n2:o, n1:o, n:o
Contexts: G:c[]
IH:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}* => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b} => exists d, {G |- d : conc b}
IH1:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}@ => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b}** => exists d, {G |- d : conc b}
H1:{a : prop}@
H2:{G |- d1 : conc a}
H3:{G, n:hyp a |- impR b1 b2 ([x] d n x) : conc (imp b1 b2)}@@
H4:{G, n:hyp a |- b1 : prop}**
H5:{G, n:hyp a |- b2 : prop}**
H6:{G, n:hyp a, n1:hyp b1 |- d n n1 : conc b2}**
H7:{G, n1:hyp b1, n:hyp a |- d n n1 : conc b2}**
H8:{G |- b1 : prop}**
H9:{G, n2:hyp b1 |- d1 : conc a}
==================================
exists d, {G |- d : conc (imp b1 b2)}
```

At this stage the application of the inner inductive hypothesis `IH1` to `H1`, `H9`, and `H7` is possible by instantiating `G` with `(G,n1:hyp b1)`, `a` with `a`, `b` with `b2`, `d1` with `d1`, and `d2` with `([x] d x n1)`. This application results in a new eigenvariable `d'` being introduced along with a new assumption `H10:{G, n1:hyp b1 |- d' n1 :  conc b2}`. We now construct an inhabitant of the type (*conc* (*imp* b_1 b_2)) using **`exists`** to instantiate the existential quantifier in the goal formula with the term (*impR* b_1 b_2 ($\lambda x.\, d'\ x$)). The resulting state is the following.

```
Vars: d': o -> o, d:o -> o -> o, b1:o, b2:o, d1:o, a:o
Nominals: n2:o, n1:o, n:o
Contexts: G:c[]
IH:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}* => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b} => exists d, {G |- d : conc b}
IH1:ctx G:c. forall a, forall b, forall d1, forall d2,
        {a : prop}@ => {G |- d1 : conc a} =>
            {G, n:hyp a |- d2 n : conc b}** => exists d, {G |- d : conc b}
H1:{a : prop}@
H2:{G |- d1 : conc a}
H3:{G, n:hyp a |- impR b1 b2 ([x] d n x) : conc (imp b1 b2)}@@
H4:{G, n:hyp a |- b1 : prop}**
H5:{G, n:hyp a |- b2 : prop}**
H6:{G, n:hyp a, n1:hyp b1 |- d n n1 : conc b2}**
H7:{G, n1:hyp b1 |- d n n1 : conc b2}**
H8:{G |- b1 : prop}**
H9:{G, n2:hyp b1 |- d1 : conc a}
H10:{G, n1:hyp b1 |- d' n1 : conc b2}
==================================
{G |- impr b1 b2 ([x] d' x) : conc (imp b1 b2)}
```

There still remains one step before **`search`** can successfully construct a derivation for this goal. Clearly *atm-app-R* would be used to derive this subgoal, and would require derivations then of $\{G \vdash b_1 : prop\}$, $\{G \vdash b_2 : prop\}$, and $\{G \vdash \lambda x.\, d'\ x : \Pi x{:}hyp\ b_1.\, conc\ b_2\}$. The first and last of these can be ensured in the current state through applications of *id* with `H8` and `H10` respectively, but a derivation for the second cannot be constructed automatically. However, similar to the earlier application of strengthening, we can apply `strengthen` to `H5` which will result in extending the assumptions with a formula of the needed form, and this case of the cut admissibility proof can be completed using `search`.

6.4 The POPLmark Challenge

Adelfa is also able to handle larger developments, such as Problem 1A of the POPLmark Challenge [ABF^{+}05]. Specifically, for this problem we show the transitivity of System $F_{<:}$. There are many existing solutions to the challenge problem, which makes it a good candidate for comparisons across systems. These also exists a solution to this problem formalized in Twelf [ARCH], and the formalization in Adelfa is based on this solution.[1] This example demonstrates more sophisticated use of contexts in reasoning than we have seen thus far, and the proof requires a more complex inductive structure. Figure 6.5 contains the specification of System $F_{<:}$ and the subtyping relation. The type *ty* is used to encode the types of the system while *sub* encodes the subtyping relation between two types. Since subtyping assumptions are necessary in the encoding of these types, we further introduce the type *bound* for this purpose.

The goal in this example is to show that if under a context Γ, S is a subtype of Q and Q is a subtype of T, then under this same context Γ, S is a subtype of T. In this proof we will need to make use of narrowing, which states that if P is a subtype of Q and under a context containing $sub\ x\ Q$ the type N is a subtype of M, then N is also a subtype of M under the same context except with assumption $sub\ x\ P$ replacing $sub\ x\ Q$. These are proved simultaneously by mutual induction on the structure of Q, proving first transitivity using narrowing for types smaller than Q and then proving narrowing using transitivity for

[1] A Twelf formalization has been presented in [Pie07] and it has been claimed that it also solves this challenge problem. However, this claim is false: the formulation of the typing calculus differs from the one presented in [ABF^{+}05] and that formulation in fact assumes away the essential aspects of the challenge [Nad19].

$$
\begin{array}{rcl}
\mathit{ty} & : & \mathrm{Type}. \\
\mathit{top} & : & \mathit{ty}. \\
\mathit{arrow} & : & \mathit{ty} \to \mathit{ty} \to \mathit{ty} \\
\mathit{all} & : & \mathit{ty} \to (\mathit{ty} \to \mathit{ty}) \to \mathit{ty} \\
\mathit{bound} & : & \mathit{ty} \to \mathit{ty} \to \mathrm{Type} \\
\mathit{sub} & : & \mathit{ty} \to \mathit{ty} \to \mathrm{Type} \\
\textit{sa-top} & : & \Pi S{:}\mathit{ty}.\, \mathit{sub}\ S\ \mathit{top} \\
\textit{sa-refl-tvar} & : & \Pi X{:}\mathit{ty}.\, \Pi U{:}\mathit{ty}.\, \Pi A{:}\mathit{bound}\ X\ U.\, \mathit{sub}\ X\ X \\
\textit{sa-trans-tvar} & : & \Pi X{:}\mathit{ty}.\, \Pi U1{:}\mathit{ty}.\, \Pi U2{:}\mathit{ty}.\, \Pi A1{:}\mathit{bound}\ X\ U1.\, \Pi A2{:}\mathit{sub}\ U1\ U2. \\
 & & \quad \mathit{sub}\ X\ U2 \\
\textit{sa-arrow} & : & \Pi S_1{:}\mathit{ty}.\, \Pi S_2{:}\mathit{ty}.\, \Pi T_1{:}\mathit{ty}.\, \Pi T_2{:}\mathit{ty}.\, \Pi a_1{:}\mathit{sub}\ T_1\ \mathit{app} S_1.\, \Pi a_2{:}\mathit{sub}\ S_2\ T_2. \\
 & & \quad \mathit{sub}\ (\mathit{arrow}\ S_1\ S_2)\ (\mathit{arrow}\ T_1\ T_2) \\
\textit{sa-all} & : & \Pi S_1{:}\mathit{ty}.\, \Pi S_2{:}\mathit{ty} \to \mathit{ty}.\, \Pi T_1{:}\mathit{ty}.\, \Pi T_2{:}\mathit{ty} \to \mathit{ty}.\, \Pi a_1{:}\mathit{sub}\ T_1\ S_1. \\
 & & \quad \Pi a_2{:}(\Pi w{:}\mathit{ty}.\, \Pi y{:}\mathit{bound}\ w\ T_1.\, \mathit{sub}\ (S_2\ w)\ (T_2\ w))). \\
 & & \quad \mathit{sub}\ (\mathit{all}\ S_1\ (\lambda x.\, S_2\ x))\ (\mathit{all}\ T_1\ (\lambda x.\, T_2\ x))
\end{array}
$$

Figure 6.5: LF Specification for System $F_{<:}$

the type Q. We have not yet specified the particular form of the contexts in this reasoning, which has some complexity due to the use of LF for the encoding. Before we can state precisely the Adelfa theorem we must define a schema for the contexts we wish to reason about.

At the outset, it is clear that in reasoning about subtyping the contexts must contain pairs of variable and subtyping assumptions of the form $(x : tm, y : sub\ x\ T)$. However, for narrowing we need to be able to identify a variable from an arbitrary location within the context and move it to the end of the context so that narrowing can be applied. This is a complication because in LF the ordering of bindings within a context must respect dependencies, and the type T in a subtyping assignment $(bound\ x\ T)$ for a variable x might well depend on some variable declared earlier in the context and such variables could not be declared within this context at arbitrary locations. In the formalization of the theorem in the Twelf system [ARCH], this issue is addressed by carefully constructing the contexts such that variables and their subtyping assumptions can be separated within a context without becoming dissociated. We follow this approach in Adelfa and introduce two new types for the purpose, $var : ty \to \text{Type}$ and $bound\_var : \Pi X{:}ty.\, \Pi T{:}ty.\, bound\ X\ T \to var\ X \to \text{Type}$ to identify the variables independently of their subtyping assumption and to ensure that any variable introduced into the context has also a single associated subtyping assumption. We then define the context schema as one which comprises three block definitions, one which keeps the variable together with its subtyping assumption, and one for each of the split collection of assumptions. These block definitions are detailed below.

$$\{T : o\}w : ty, x : \text{var}\ w, y : bound\ x\ T, z : bound\_var\ w\ x\ y$$

$$x : ty, y : \text{var}\ w$$

$$\{V : o, T : o, DV : o\}x : bound\ V\ T, y : bound\_var\ V\ T\ x\ DV$$

Let this context schema be given the identifier c.

The theorem we prove in Adelfa is the following.

$$\Pi\, G : c.\forall q : o.\, \{G \vdash q : ty\} \supset$$

$$(\Pi\, L : c.\forall s : o.\forall t : o.\forall d_1 : o.\forall d_2 : o.$$

$$
\begin{aligned}
&\qquad\qquad \{L \vdash d_1 : \mathit{sub}\ s\ q\} \supset \{L \vdash d_2 : \mathit{sub}\ q\ t\} \supset \exists d_3. \{L \vdash d_3 : \mathit{sub}\ s\ t\}) \\
&\wedge \\
&(\Pi\, L : c.\forall x : o.\forall t_1 : o.\forall t_2 : o.\forall p : o.\forall d_1 : o.\forall d_2 : o \to o.\forall dv : o. \\
&\qquad\qquad \{L \vdash d_1 : \mathit{sub}\ p\ q\} \supset \\
&\qquad\qquad \{L, n_1 : \mathit{bound}\ x\ q, n_2 : \mathit{bound\_var}\ x\ q\ n_1\ dv \vdash d_2\ n_1 : \mathit{sub}\ t_1\ t_2\} \supset \\
&\qquad\qquad \exists d_4 : o \to o. \\
&\qquad\qquad\qquad \{L, n_1 : \mathit{bound}\ x\ p, n_2 : \mathit{bound\_var}\ x\ p\ n_1\ dv \vdash d_4\ n_1 : \mathit{sub}\ t_1\ t_2\}).
\end{aligned}
$$

The outer induction of the proof is on $\{G \vdash q : ty\}$. Within this proof, transitivity is shown by induction on $\{L \vdash d_1 : \mathit{sub}\ s\ q\}$ and narrowing is shown by induction on the formula $\{L, n_1 : \mathit{bound}\ x\ q, n_2 : \mathit{bound\_var}\ x\ q\ n_1\ dv \vdash d_2\ n_1 : \mathit{sub}\ t_1\ t_2\}$. As in the admissibility of cut, the management of the context ordering is realized in the Adelfa development through the application of the tactics `weaken`, `strengthen`, and `permutectx`. With this understanding of the encoding and the structure of the proof, it is easy to visualize how the formal proof should be constructed. The detailed development that makes all the steps explicit is available from the Adelfa web site and so we will not present the formalization in detail here.

Chapter 7

Comparisions with Related Work

The usefulness of LF as a specification language can be seen through the wide variety of tasks in which it has been used since its introduction, for example [BC04, LCH07, Ler09]. Reasoning about such LF specifications has thus also been of interest and there exist various approaches which attempt to provide a means for effectively reasoning about systems through their encoding in LF. In this chapter we consider existing systems and approaches to reasoning about specifications written in LF and contrast them with the logic and proof system developed in this thesis.

7.1 Twelf

The Twelf system is a well established system which can be used to reason about LF specifications [PS99, PS02]. It has a wide variety of existing examples, many available from the Twelf website. In this system, no distinction is made between the specification and the reasoning level. The approach taken by Twelf is a computational view of reasoning where properties of the object system are also encoded as LF types, with constructors for these types essentially being proof scripts. An external analysis is used to check properties of this encoding and from this results about the specification are extracted. In particular, typically the LF type encoding a property of interest is shown to be total through the use of coverage and termination checking.

Due to this view of reasoning, the sort of statements which can be checked in Twelf are all of functional structure; some collection of derivations are given as input and some other collection of derivations are constructed as output. Thus, only formulas of a $\forall\exists$ form are checkable in Twelf which limits the expressiveness of reasoning in this way. Reasoning in Twelf also does not construct proofs explicitly, instead the validity of these formulas are extracted from the specification along with a positive result from the totality checking procedure.

Another aspect to consider in this comparison is the treatment of contexts in Twelf reasoning. Twelf permits definitions similar to context schemas, though they are called regular worlds in this setting, but the contexts are kept implicit in the reasoning process. Thus there is a single implicit context across an instance of the analysis, which limits the flexibility of the system. Properties which require typing derivation in different contexts cannot be expressed in Twelf. Keeping the contexts implicit also obscures the true structure of the analysis performed by the system.

7.2 The Logic $\mathcal{M}_2^+$

The logics $\mathcal{M}_2$ [SP98] and $\mathcal{M}_2^+$ [Sch00] were developed to formalize reasoning as performed in Twelf. The logic provides a logical foundation to the sort of reasoning performed by Twelf which can address some of the shortcomings of using an external process like totality checking for reasoning. However, because they are focused specifically on capturing reasoning as it is viewed in Twelf, the logics $\mathcal{M}_2$ and $\mathcal{M}_2^+$ have a different flavor than the logic presented in this thesis. These logics are based on an understanding of Twelf rather than directly on understanding derivability and reasoning in LF, and does not illuminate the structure of reasoning steps as we aim to do. This view of reasoning leads to a more complex form of induction which is realized through proof terms corresponding to recursive total functions. Thus, while the logic $\mathcal{M}_2^+$ provides a formalization for Twelf reasoning it is still rooted in the computational approach to reasoning used by Twelf of extracting proofs through checking totality of functions of dependent types.

Due to the approach of its design, the logic $\mathcal{M}_2^+$ also has the same limitations in expressivity as we have seen in Twelf. Proofs in this logic are functions, and the logic ensures that only total functions can be constructed. Properties which are not expressed as function types cannot then be reasoned about with this logic. The logic also maintains a single context across a formula, thus again restricting the ability to express relations over derivations in LF which use different contexts.

7.3 Beluga

Another well established system for reasoning about LF is Beluga [PD10]. Beluga uses a two level approach, providing a dependently typed functional programming language on top of LF. The reasoning level of Beluga is based on contextual modal type theory [NPP08] which allows an explicit treatment of contexts in typing. Beluga is, like Twelf, based on a computational view of reasoning. In Beluga one writes dependently-typed recursive functions and the type system ensures that the admitted programs are ones which are total and thus encode sound proofs. These functions are not proofs in themselves, as the type checking is critical to reasoning and only together do they constitute a proof. A distinguishing feature of Beluga is that it is intended as a programming language and so provides the ability to execute functions written in this way.

The use of contextual modal type theory as the reasoning level permits a more expressive syntax for formulas than is available in Twelf, in particular the role of contexts is much more expressive in Beluga. However, the properties which can be encoded as types in the system remain restricted to the same functional structure relating some set of input derivations to a set of output derivations. Reasoning in Beluga also relies on some external analysis, in this case type checking, and so proofs are not constructed directly as can be done for the logic we have defined.

7.4 Harpoon

Harpoon [EJP20] is an interactive tool which assists in the construction of Beluga functions using a collection of tactics for generating proof scripts which then generate valid Beluga functions. Programming in Beluga in this way provides the feel of theorem proving and many of the tactics have a similar feel to those which we include in Adelfa from Chapter 5.

Harpoon however, determines the structure and collection of these tactics from an understanding of Beluga programming rather than from inspection of how reasoning proceeds in LF These derivations are also not proofs directly, as they remain dependent on type checking of Beluga to constitute a proof. The expressivity of Harpoon reasoning also remains limited by the underlying mapping into Beluga functions.

7.5 Cocon

There has recently been some interesting work done in developing the Cocon Type Theory [PAF+19, PS20]. Cocon is a powerful type theory which unifies LF methodology with dependent type theories such as Martin-Löf Type Theory. A system based on this type theory would be quite powerful for reasoning about dependently-typed specifications, but to our knowledge such an implementation has not yet been explored. It would be interesting to consider a system based on Cocon because of the ability to mix representations and computations in the type system, but this is a consideration which is outside the scope of current work.

7.6 Abella-LF

A different view of reasoning about LF derivability is taken in [SC14], which utilizes a translation-based approach. Reasoning is realized through translating LF judgements into relations in a predicate logic and reasoning is performed over the translated form. The result of derivations constructed over the translation are also lifted to LF using the translation. This approach has been implemented as Abella-LF, a variant of the Abella Prover [Gac08, Gac09a].

Abella uses a two-level approach to reasoning where derivability is encoded as a definition in the reasoning logic [Gac09b]. Abella-LF builds in a translation and decoding of LF terms using syntactic sugar on Abella terms to provide users with an illusion of working directly with LF. The Abella system is very expressive, and in fact may be too expressive for reasoning about LF specifications as it is unclear what meaning reasoning about the translation has in LF. The tactics available in Abella are not based on conceptual steps of LF reasoning and so developments in Abella-LF do not always correspond with reasoning steps of LF. Because of this, it is possible for the translation to be exposed during reasoning and the illusion of working in an LF setting to be lost. This is especially apparent in the treatment of contexts; the behaviour of Abella-LF does not align with intuition obtained from understanding LF derivability.

The translation underlying Abella-LF is also not correct in the form it is used; the derivability of a translated LF judgement does not ensure the derivability of the original LF judgement unless restricted to the Canonical LF system. This issue will have to be addressed for it to be possible to make claims about the lifting of any results to LF. Our work provides an understanding of derivability in LF which could be used as a foundation for an implementation of Abella-LF which addresses these issues. Specifically, having identified proof rules capturing specific and well-understood reasoning steps we could use this collection to constrain reasoning about the translation in a way which will be meaningful in the LF setting.

Chapter 8

Conclusion

In this thesis we have considered the construction of a logic for reasoning about LF specifications. The logic we have developed is based on a semantical approach using a substitution based interpretation of quantification and relying on checking the derivability of LF judgements represented by the atomic formulas. This logic is also given a corresponding proof system which formalizes arguments based on the semantics. This proof system builds in an understanding of LF typing judgements, incorporates a means for arguing inductively based on the heights of derivations for such judgements and encodes meta-theoretic observations about LF derivability through axioms that reflect the contents of these observations. We have also mechanized the construction of proofs in the proof system in the proof assistant Adelfa. The usefulness of this implementation, and by extension the proof system, has been shown through a collection of examples of reasoning about a variety of object systems.

The logic and the proof system that we have developed for it display a fair amount of flexibility. For example, it is possible to express in the logic all the properties that are expressible in a system like Twelf [PS99] or Beluga [PD10]. Further, it is possible to construct proofs for such properties using the proof system. Finally, going beyond these systems, the logic allows for the expression of properties that have a disjunctive character or that embody an alternation of term-level universal and existential quantifiers and formal proofs for such properties can also be constructed. There are, nevertheless, particular aspects of the logic and the proof system that we would like to enhance or improve so as to provide greater flexibility in reasoning. We discuss some of these aspects below as avenues for further research.

8.1 Encoding Properties of Contexts in the Logic

The first extension we consider is encoding properties of contexts in the logic. Such properties would permit reasoning about relations between one or more contexts, such as subsumption of context schemas. These relations would extend the reasoning capabilities of the logic through the ability to make use of properties of the contexts appearing in formulas.

One property of this sort might be context schema subsumption, allowing context expressions known to satisfy one context schema to also be identified as satisfying any other schema which subsumes it. Such a property could be used to lift a context expression satisfying the more restrictive schema to the more expressive schema in order to apply, for example, lemmas proved about contexts of the more general form.

These relations may also take other forms, for example they might identify contexts which arise from derivations for distinct types in LF pertaining to different relations involving a particular term. For example, suppose that we define a reduction relation between STLC terms which holds for terms t_1 and t_2 if t_2 is obtained from t_1 by reducing a β-redex somewhere in the term. This might be encoded by the following LF declarations.

$$\begin{array}{lll}
\mathit{reduce} & : & \mathit{tm} \to \mathit{tm} \to \mathrm{Type}. \\
\mathit{red\text{-}beta} & : & \Pi T{:}\mathit{tp}.\, \Pi R{:}\mathit{tm} \to \mathit{tm}.\, \Pi M{:}\mathit{tm}.\, \Pi M'{:}\mathit{tm}.\, \mathit{reduce}\ (R\ M)\ M' \to \\
 & & \qquad \mathit{reduce}\ (\mathit{app}\ (\mathit{lam}\ T\ (\lambda x.\, R\ x))\ M)\ M' \\
\mathit{red\text{-}lam} & : & \Pi T{:}\mathit{tp}.\, \Pi R{:}\mathit{tm} \to \mathit{tm}.\, \Pi R'{:}\mathit{tm} \to \mathit{tm}.\, \Pi x{:}\mathit{tm}.\, \mathit{reduce}\ (R\ x)\ (R'\ x) \to \\
 & & \qquad \mathit{reduce}\ (\mathit{lam}\ T\ (\lambda x.\, R\ x))\ (\mathit{lam}\ T\ (\lambda x.\, R'\ x)) \\
\mathit{red\text{-}app\text{-}1} & : & \Pi M{:}\mathit{tm}.\, \Pi N{:}\mathit{tm}.\, \Pi M'{:}\mathit{tm}.\, \mathit{reduce}\ M\ M' \to \\
 & & \qquad \mathit{reduce}\ (\mathit{app}\ M\ N)\ (\mathit{app}\ M'\ N) \\
\mathit{red\text{-}app\text{-}2} & : & \Pi M{:}\mathit{tm}.\, \Pi N{:}\mathit{tm}.\, \Pi N'{:}\mathit{tm}.\, \mathit{reduce}\ N\ N' \to \\
 & & \qquad \mathit{reduce}\ (\mathit{app}\ M\ N)\ (\mathit{app}\ M\ N')
\end{array}$$

Contexts relevant to derivations of *reduce* will have the structure given by a single block schema $(z : \mathit{tm})$, as we can see from the type of *red-lam*. We can see that for a particular term t, we might define a relation which holds between a context expression satisfying the schema for typing in the STLC we have seen previously, $\{T : o\}(x : tm, y : of\ x\ T)$, used in typing t in the STLC, and a context expression satisfying this reduction schema for reducing t, which includes only the instances of $x : tm$ from the typing context. Thus we could express formulas in the logic which relate derivations using different context expressions in a way which permits the structure of some contexts to be informed by others. For example, subject reduction might be stated as the following theorem where ctx-rel represents the relation described above and ty-ctx and red-ctx denote the STLC typing and reduction context schemas respectively.

$$\begin{array}{l}\Pi\, G_1 : \text{ty-ctx}.\Pi\, G_2 : \text{red-ctx}.\forall M : o.\forall N : o.\forall T : o.\forall D_1 : o.\forall D_2 : o. \\ \qquad \text{ctx-rel}\ G_1\ G_2 \supset \{G_1 \vdash D_1 : of\ M\ T\} \supset \{G_2 \vdash D_2 : reduce\ M\ N\} \supset \\ \qquad\qquad\quad \exists D_3 : o. \{G_1 \vdash D_3 : of\ N\ T\}\end{array}$$

At the outset we identify two things which will be needed to realize this extension to the logic:

1. A means to express the properties of contexts in the logic and
2. The ability to interpret the properties of contexts inductively.

We might take inspiration from the context definitions of Abella for describing arbitrary properties of contexts within the logic, and think of building into the proof system an ability to reason inductively about contexts based on the definitions.

8.2 Schematic Polymorphism

Another interesting direction is to introduce a form of polymorphism called schematic polymorphism, as has been done for Abella in [NW18]. This form of polymorphism would simplify construction of arguments which have the same structure regardless of type instance, reducing repetition in reasoning about systems which require structures to be repeated at many types within an encoding.

For example, we might encode in LF an *append* relation for lists of integers. Properties such as the functionality of *append* have a structure which does not make use of the knowledge that the elements in the list are integers, and thus would have the same structure for lists containing elements of any other type we might include in the specification. Since this proof is general with respect to the particular element type, it would be useful to construct the proof of this property once and use it for proving the functionality of append for other sorts of lists such as lists of pairs of integers or lists of booleans. To realize this in Adelfa we will need to extend both the declaration syntax and the proof system with relations and proof rules that are generic with respect to a particular type. These schematic type families and proofs would provide a means of expressing structure that is identical regardless of a particular instance.

For the LF declaration syntax, we would require a means for describing a collection of type families which follow a particular structure. Thus we might permit schematic type variables in these declarations, and any particular LF declaration could be generated by replacing this schematic variable with a ground type in LF. For example we might describe the *append* relation using something like the following.

$$\begin{array}{lll}
\mathit{append}_A & : & \mathit{list}_A \to \mathit{list}_A \to \mathit{list}_A \to \mathrm{Type}. \\
\mathit{app\text{-}nil} & : & \Pi L{:}\mathit{list}_A.\, \mathit{append}\ \mathit{nil}\ L\ L. \\
\mathit{app\text{-}cons} & : & \Pi X{:}A.\, \Pi L_1{:}\mathit{list}_A.\, \Pi L_2{:}\mathit{list}_A.\, \Pi L_3{:}\mathit{list}_A. \\
 & & \qquad \mathit{append}\ L_1\ L_2\ L_3 \to \mathit{append}\ (\mathit{cons}\ X\ L_1)\ L_2\ (\mathit{cons}\ X\ L_3).
\end{array}$$

In this description, A is a schematic variable which can be replaced with any particular ground LF type to generate that particular instance of these declarations encoding the append relation for lists of that sort.

In the logic, we would extend the formula syntax by permitting quantification over schematic type variables. Derivations in the proof system for such formulas would describe schematic proofs which are invariant under the choice of type instance, and will require new proof rules describing sound reasoning steps of this kind. The observation underlying these proof rules will be that the derivations should not identify particular instances of the schematic type variables, and so must be such that the proofs which can be constructed are the same for every type. For example, we might state the functionality of append in a generalized form as the following.

$$\begin{array}{l}\forall L_1 : o.\forall L_2 : o.\\ \qquad \{\cdot \vdash L_1 : \mathit{list}_A\} \supset \{\cdot \vdash L_2 : \mathit{list}_A\} \supset \exists L_3 : o.\exists D : o.\{\cdot \vdash D : \mathit{append}_A\ L_1\ L_2\ L_3\}\end{array}$$

A proof for this formula in the refined proof system would be a schematic proof as all instances must have the same structure regardless of the particular instance for the generic type A. Any explicit instance of the proof could then be generated as needed by choosing a particular ground type to instantiate the schematic variable.

Bibliography

[ABF$^+$05] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized metatheory for the masses: The POPLmark challenge. In *Theorem Proving in Higher Order Logics: 18th International Conference*, number 3603 in LNCS, pages 50–65. Springer, 2005.

[ARCH] Michael Ashley-Rollman, Karl Crary, and Robert Harper. POPLmark group from CMU's solution. `https://www.seas.upenn.edu/~plclub/poplmark/cmu.html`.

[BC04] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. Springer, 2004.

[BCG$^+$14] David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2), 2014.

[EJP20] Jacob Errington, Junyoung Clare Jang, and Brigitte Pientka. Mechanizing Metatheory Interactively. In *Fifteenth Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, 2020.

[Gac08] Andrew Gacek. The Abella interactive theorem prover (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *Fourth International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 154–161. Springer, 2008.

[Gac09a] Andrew Gacek. The Abella system and homepage. `http://abella.cs.umn.edu/`, 2009.

[Gac09b] Andrew Gacek. *A Framework for Specifying, Prototyping, and Reasoning about Computational Systems.* PhD thesis, University of Minnesota, 2009.

[GMN11] Andrew Gacek, Dale Miller, and Gopalan Nadathur. Nominal abstraction. *Information and Computation*, 209(1):48–73, 2011.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

[HL07] Robert Harper and Daniel R. Licata. Mechanizing metatheory in a logical framework. *Journal of Functional Programming*, 17(4–5):613–673, July 2007.

[LCH07] Daniel K. Lee, Karl Crary, and Robert Harper. Towards a mechanized metatheory of standard ML. In *POPL '07: Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 173–184, New York, NY, USA, 2007. ACM Press.

[Ler09] Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *J. of Logic and Computation*, 1(4):497–536, 1991.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.

[MN87] Dale Miller and Gopalan Nadathur. A logic programming approach to manipulating formulas and programs. In Seif Haridi, editor, *IEEE Symposium on Logic Programming*, pages 379–388, San Francisco, September 1987.

[Nad19] Gopalan Nadathur. Personal communication, August 2019.

[NL05] Gopalan Nadathur and Natalie Linnell. Practical higher-order pattern unification with on-the-fly raising. In *ICLP 2005: 21st International Logic Programming Conference*, volume 3668 of *LNCS*, pages 371–386, Sitges, Spain, October 2005. Springer.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual model type theory. *ACM Trans. on Computational Logic*, 9(3):1–49, 2008.

[NW18] Gopalan Nadathur and Yuting Wang. Schematic polymorphism in the Abella proof assistant. *CoRR*, abs/1806.07523, 2018.

[PAF+19] Brigitte Pientka, Andreas Abel, Francisco Ferreira, David Thibodeau, and Rebecca Zucchini. Cocon: Computation in Contextual Type Theory. 2019.

[PD10] Brigitte Pientka and Joshua Dunfield. Beluga: A framework for programming and reasoning with deductive systems (system description). In J. Giesl and R. Hähnle, editors, *Fifth International Joint Conference on Automated Reasoning*, number 6173 in LNCS, pages 15–21, 2010.

[PE88] Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *Proceedings of the ACM-SIGPLAN Conference on Programming Language Design and Implementation*, pages 199–208. ACM Press, June 1988.

[Pie07] Brigitte Pientka. Proof pearl: The power of higher-order encodings in the logical framework LF. In Klaus Schneider and Jens Brandt, editors, *Theorem Proving in Higher Order Logics, 20th International Conference, TPHOLs 2007, Kaiserslautern, Germany, September 10-13, 2007, Proceedings*, volume 4732 of *Lecture Notes in Computer Science*, pages 246–261. Springer, 2007.

[PS99] Frank Pfenning and Carsten Schürmann. System description: Twelf — A meta-logical framework for deductive systems. In H. Ganzinger, editor, *16th Conf. on Automated Deduction (CADE)*, number 1632 in LNAI, pages 202–206, Trento, 1999. Springer.

[PS02] Frank Pfenning and Carsten Schürmann. *Twelf User's Guide*, 1.4 edition, December 2002.

[PS20] Brigitte Pientka and Ulrich Schöpp. Semantical analysis of contextual types. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures*, pages 502–521, Cham, 2020. Springer International Publishing.

[SC14] Mary Southern and Kaustuv Chaudhuri. A Two-Level Logic Approach to Reasoning About Typed Specification Languages. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 557–569, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Sch00] Carsten Schürmann. *Automating the Meta Theory of Deductive Systems*. PhD thesis, Carnegie Mellon University, October 2000. CMU-CS-00-146.

[SP98] Carsten Schürmann and Frank Pfenning. Automated theorem proving in a simple meta-logic for LF. In Claude Kirchner and Hélène Kirchner, editors, *15th Conf. on Automated Deduction (CADE)*, volume 1421 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 1998.

[Tiu06] Alwen Tiu. A logic for reasoning about generic judgments. In A. Momigliano and B. Pientka, editors, *Int. Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP'06)*, volume 173 of *ENTCS*, pages 3–18, 2006.

[WCPW03] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Carnegie Mellon University, 2003. Revised, May 2003.